Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Programové ovládání VF generátoru a měřiče

Control of VF Generator and Measuring Unit

2014 Lukáš Stehlík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Lukáš Stehlík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Programové ovládnání VF generátoru a měřiče
Control of VF Generator and Measuring Unit

Zásady pro vypracování:

Pro zařízení s VF generátorem a měřičem, určené pro testování VF odolnosti elektronických výrobků, navrhněte galvanicky oddělené komunikační rozhraní. Komunikace se zařízením musí být přes síťové rozhraní Ethernet a při návrhu upřednostněte výrobky MOXA a Papouch. Pro ovládání zařízení napište program v jazyce C#.

1. Vyberte vhodné komponenty a navrhněte galvanicky oddělené komunikační rozhraní.
2. Realizujte a ověřte funkcionalitu navrženého zapojení. Vytvořte potřebnou dokumentaci zapojení.
3. Navrhněte uživatelské rozhraní řídícího programu a sepište požadavky na jeho funkcionalitu.
4. Napište program a proveďte ověření jednotlivých funkcí programu provedením měření.
5. Ověřte výsledky svých měření s výsledky staršího programu. Porovnejte a zdůvodněte případné odchylky.
6. Vyhodnoťte spolehlivost, rychlost a stabilitu navrženého řešení.

Seznam doporučené odborné literatury:

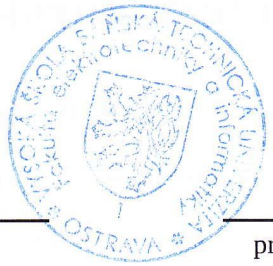
1. Signal Generator PMM 3000 10kHz-1GHz, Operating manual, dokumentace výrobce
2. Power Meter PMM 6600 10kHz-1GHz, Operating manual, dokumentace výrobce
3. Signal Generator SG2000, Operating manual, dokumentace výrobce
4. Interface RS485 PMM 6485, dokumentace výrobce
5. http://www.papouch.com, datové listy vybraných komponent
6. http://www.moxa.cz, datové listy vybraných komponent

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Petr Olivka**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 ..... Vlastík Lukáš .....

Na tomto místě bych chtěl poděkovat vedoucímu práce panu Ing. Petru Olivkovi za udělené rady a připomínky při tvorbě této práce, firmě ZAM-Servis s.r.o. za návrh tématu a prostředky k uskutečnění práce a ostatním, kteří mi pomohli nebo mě podpořili v práci.

Abstrakt

Bakalářská práce se zabývá komunikací mezi sestavou měřících přístrojů a řídícím počítačem s využitím Ethernetové počítačové sítě. Měřící sestava je určena pro testování odolnosti proti rušení šířeným vedením a indukovanými vysokofrekvenčními poli. Sestava je složena z vysokofrekvenčního generátoru, měřiče výkonu, vazebních členů a zkoušeného zařízení. Práce se dále zabývá galvanickým oddělením sestavy měřících přístrojů od řídícího počítače za využití optického Ethernetového kabelu. Výsledkem práce jsou výstupy z vývoje navrženého zapojení a řešení, praktického ověřování funkčnosti, zhodnocení výsledků měření a praktického použití.

Klíčová slova: C#, vysokofrekvenční generátor, měřič výkonu, vysokofrekvenční odolnost, virtuální sériový port, RS232, RS485

Abstract

The aim of this bachelor thesis is to solve communication between setup of measuring instruments and controlling computer via computer network. Setup of measuring instruments is designated for testing electromagnetic compatibility and setup consists of wide-band rf signal generator, rf power meter, coupling/decoupling networks and equipment under testing. Further focus of thesis is developement of galvanic isolation using optical fiber cable. The results of thesis are final wiring of galvanic isolation, practical results from testing, evaluated results of measuring and example of practical application.

Keywords: C#, wide-band rf signal generator, rf power meter, Electromagnetic Compatibility virtual serial port, RS232, RS485

Seznam použitých zkratek a symbolů

Bd	–	baud – přenosová rychlost
CSMA/CD	–	Carrier Sense Multiple Access with Collision Detection
ČSN EN	–	Česká Státní Norma Evropská Norma
EMC	–	Elekromagnetická kompatibilita
HMI	–	Human-machine interface
IEEE	–	Institute of Electrical and Electronics Engineers
LAN	–	Local Area Network
Mbit	–	megabit
MiniGBIC	–	mini gigabit interface converter
SFP	–	Small Form-factor Pluggable
UTP	–	Unshielded Twisted Pair
V	–	volt
VSP	–	Virtuální sériový port
dB	–	decibel
vf	–	vysokofrekvenční

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato práce postupně popisuje technologie, komponenty sestavy, realizaci a vývoj galvanického oddělení měřící sestavy, dostupnosti měřící sestavy pomocí počítačové sítě Ethernet a programového ovládání generátorů a měřiče vysokofrekvenční energie využívaných při zkoušce kompatibility EMC. K závěru práce jsou uvedeny výsledky z ověřování funkcionality programového ovládaní, výsledky z měření a praktické použití.

1.1 Podnět pro práci

Elektronická zařízení mohou být rušena signály šířícími se po napájecích vodičích, signálových vedeních a zemnících spojeních. Signály jsou zásadně ve formě elektromagnetického pole přicházejícího z úmyslných vf vysílačů v kmitočtovém rozsahu od 150 kHz do 80 MHz, které může působit na celou délku kabelů připojených k instalovanému zařízení. Přívodní a výstupní pohyblivé přívody jako síťové přívody, komunikační vedení a propojovací kabely se chovají jako pasivní přijímací anténní sítě, jejichž délka může být několikanásobkem délky vlny. Zařízení je tedy vystaveno proudům protékajících přes zařízení mezi těmito kabelovými sítěmi. Citlivá zařízení se testují na odolnost proti šířeným vedením a vf indukovanými poli podle normy ČSN EN 6000-4-6 „Odolnost proti rušením šířeným vedením, indukovaným vysokofrekvenčními poli“, která definuje testování elektromagnetické odolnosti, parametry testovací sestavy a stupně přísnosti pro zařízení.

Během zkoušky touto metodou je zkoušené zařízení vystaveno zdroji rušení zahrnující elektrická i magnetická pole, které modelují pole přicházející z úmyslných vf vysílačů. Tato rušivá pole jsou aproximována elektrickými a magnetickými blízkými poli, která jsou následkem napětí a proudů vytvářených zkušební sestavou. Zkušební signál při zkoušce je modulován amplitudově sinusovou vlnou 1 kHz do hloubky 80 %. Jeho zdrojem je vf generátor, který se připojuje ke zkoušenému zařízení vazebním obvodem.

Výrobce zkušebních komponent (vf generátor, měřič výkonu, zesilovač) a příslušenství dodával k přístrojům programové ovládaní komunikující přes sériovou linku s komponentami. Nereagoval však na vývoj stále nových operačních systémů Windows a programové ovládaní neaktualizoval. Vývoj operačních systémů Windows a postupné nahrazování zastaralých počítačů za nové zapříčinilo to, že původní programové ovládaní dodávané výrobcem se stalo nekompatibilním s těmito novými systémy. Programové ovládaní je nestabilní na moderních operačních systémech Windows, zamrzá, padá a není možné jej spustit na 64bitových systémech. Vznikl tedy požadavek na vývoj náhradního programového ovládaní, které bude funkční na moderních operačních systémech s přizpůsobením funkcí dle specifických požadavků.

Zároveň s požadavkem na vývoj nového programového ovládaní vznikl požadavek na realizaci galvanicky odděleného komunikačního rozhraní mezi zkušební sestavou a řídícím počítačem, kde komponenty zkušební sestavy budou dostupné přes Ethernetovou počítačovou síť. Ovládaní celé sestavy by tedy bylo možné provádět z kteréhokoliv počítače v síti LAN, tímto by odpadla nutnost připojovat pokaždé ke zkušební sestavě notebook.

2 Technologie

Kapitola Technologie stručně seznamuje čtenáře s hlavními využitými technologiemi použité v bakalářské práci.

2.1 Ethernet

Jedná se v současné době o nejrozšířenější souhrn technologií pro budování sítí typu LAN založených na principu CSMA/CD. Ethernet se stal víceméně standardem pro svou jednoduchost a nízkou cenu, tímto vytlačil ostatní alternativní technologie. Vývoj Ethernetu začal v roce 1972 v laboratořích firmy Xerox. Ethernet je standardizován pod záštitou mezinárodní organizace IEEE s označením IEEE 802.3 určující specifikace fyzické a linkové vrstvy Ethernetu. Technologie Ethernet využívá jako přenosové médiu nejčastěji kroucený pár, optický kabel a v dřívějších dobách koaxiální kabel. Ethernet je standardizován ve více verzích podle jeho vývoje a verze lze jednoduše rozdělit do těchto kategorií:

- „Původní“ Ethernet:
  - rychlost: 10 Mbit/s,
  - přenosové médium: koaxiální kabel, kroucený pár, multimodový optický kabel,
- Fast Ethernet:
  - rychlost: 100 Mbit/s,
  - přenosové médium: kroucený pár, multimodový optický kabel,
- Gigabit Ethernet:
  - rychlost: 1000 Mbit/s,
  - přenosové médium: kroucený pár, multimodový optický kabel, singlemodový optický kabel,

2.2 Optické vlákno

Optické vlákno je dielektrický vlnovod ze skla nebo ve speciálních případech z průhledného plastu, ve kterém se šíří elektromagnetické vlnění ve formě viditelného nebo infračerveného záření ve směru osy vlákna. Pro přenos elektromagnetické vlny se využívá principu totálního odrazu na rozhraní dvou prostředí s rozdílným indexem lomu, kde jádro má index lomu vyšší než jeho plášť. Aktuálně nachází optická vlákna využití převážně v telekomunikacích, kde tvoří většinu dálkových telekomunikačních sítí. Vlákna se používají místo metalických vodičů, protože signály jsou přenášeny s menší ztrátou a zároveň je optické vlákno imunní vůči elektromagnetickému rušení. Optická vlákna dělíme dle módů na:

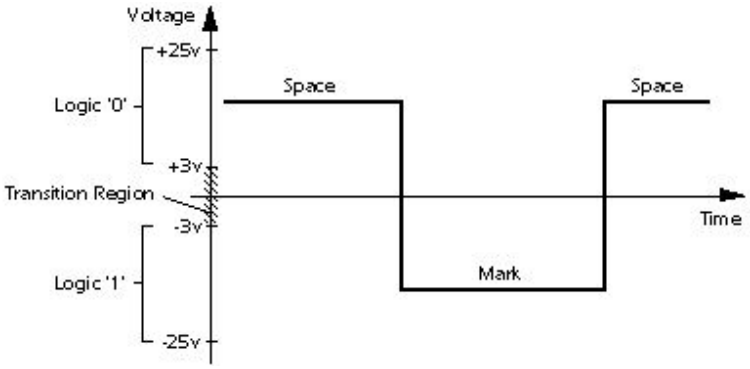
- Jednomódová vlákna (SM - singlemode):

  - vedou pouze jeden jediný paprsek,
  - pro buzení se využívá laserová dioda,
  - nevyskytuje se módová disperze,
  - průměr jádra 8 μm,
  - využití páteřní spoje, spoje na větší vzdálenosti.

- Mnohomódová vlákna (MM - multimode):
  - vedou více paprsků,
  - pro buzení se využívá LED nebo laserové diody,
  - výskyt módové disperze,
  - průměr jádra větší jak 10 μm,
  - využití v LAN sítích.

2.3 RS232

RS232 je rozhraní pro přenos informací vytvořené původně pro komunikaci mezi dvěma zařízeními do vzdálenosti 20 m, například mezi modemem (DCE) a terminálem (DTE). Rozhraní je určeno pro arytmický sériový přenos dat, tzn. že jednotlivé bity přenášených dat jsou vysílány postupně (v sérii) po jednom páru vodičů v každém směru, pomocí pevně nastavené přenosové rychlosti a synchronizace sestupnou hranou startovacího impulzu. RS232 využívá pro komunikaci dvě napěťové úrovně, kde logická 1 označována také jako „marking state“ je reprezentována zápornou úrovní napětí a logická 0 označována také jako „space state“ je reprezentována kladnou úrovní napětí. Ukázka průběhu signálů na rozhraní RS232 je zobrazena na obrázku 2.1. Obrázek převzat z článku [1].



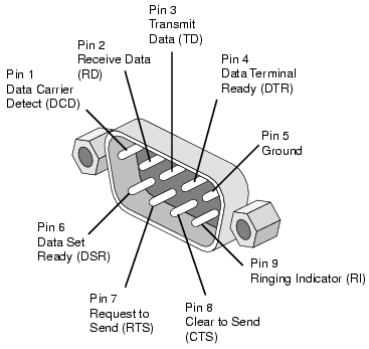
Obrázek 2.1: Datové signály RS232.

Pro generování napěťových úrovní se nejčastěji využívá napěťový zdvojovač z 5 V a invertor. Logické úrovně jsou poté reprezentovány hodnotami -10 V pro logickou 1 a +10 V pro logickou 0. Hodnoty napěťových úrovní se mohou lišit, ale musí být v mezích uvedených v tabulce 2.1.

Úroveň	Vysílač	Přijímač
logická 1	-5 V až -15 V	-3 V až -25 V
logická 0	+5 V až +15 V	+3 V až +25 V
nedefinovaný	+3 V až -3 V	

Tabulka 2.1: Napěťové úrovně RS232.

Obrázek 2.2 (obrázek převzat z článku [6]) vyobrazuje piny konektoru Cannon 9 pro rozhraní RS232 využitého v bakalářské práci. Zapojení pinů konektoru a popis signálu je znázorněn v tabulce 2.2.



Obrázek 2.2: Zapojení konektoru Cannon 9.

Pin	Název	Popis
1	DCD - Data Darrier Detect	Detekce nosné.
2	RXD - Receive Data	Tok dat z modemu do terminálu.
3	TXD - Transmit Data	Tok dat z terminálu do modemu.
4	DTR - Data Terminal Ready	Terminál signálem oznamuje modemu, že je připraven komunikovat.
5	GND - Ground	Zem.
6	DSR - Data Set Ready	Modem signálem oznamuje terminálu, že je připraven komunikovat.
7	RTS - Request To Send	Terminál signálem oznamuje modemu, že komunikační linka je volná.
8	CTS - Clear To Send	Modem signálem oznamuje terminálu, že komunikační linka je volná.
9	RI - Ring Indicator	Modem signálem oznamuje terminálu, že na telefonní lince detekoval signál zvonění.

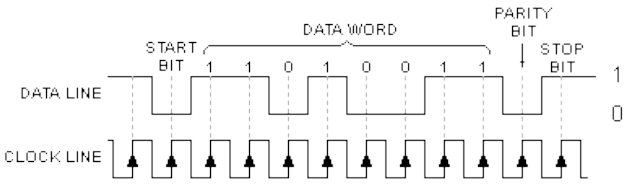
Tabulka 2.2: Přiřazení pinů konektoru Cannon 9 a popis signálů.

Rozhraní RS232 umožňuje zabezpečení přenosu pomocí parity. Parita je nejjednodušší způsob s minimálními požadavky na výpočetní výkon pro zabezpečení přenosu informací, kdy se ve vysílacím zařízení sečte počet jedničkových bitů a doplní se paritním bitem tak, aby byla dodržena domluvená parita. Parita u RS232 je následující:

- Odd parity: počet jedničkových bitů se doplní o paritní bit tak, aby celkový součet jedničkových bitů byl lichý,
- Even parity: počet jedničkových bitů se doplní o paritní bit tak, aby celkový součet jedničkových bitů byl sudý,
- Space parity: paritní bit je vždy logická 0, využití při komunikaci 7bitového zařízení s 8bitovým,
- Mark parity: paritní bit je vždy logická 1,

Rámec RS232 je kompletní přenosová skupina, tzn. přenášená data (7 nebo 8bitová) doplněná o Start bit, Stop bit a paritu. Konec rámce je definován pomocí Stop bitu, který zároveň zajišťuje prodlevu pro přijímač. Zdvojený Stop bit se používá u pomalejších zařízení pro doběh zpracování přijatého znaku, jedná se o standard na 110 Bd. Každý přenesený byte konstantní rychlostí je nutno synchronizovat pomocí sestupné hrany Start bitu, za kterou již následují posílaná data. Obrázek 2.3 (obrázek převzat z článku [1]) ilustruje synchronizaci a na obrázku 2.4 (obrázek převzat z článku [1]) je vyobrazen rámec RS232.

Obrázek 2.3: Synchronizace RS232.



Obrázek 2.4: Rámec RS232.

Maximální délka vedení pro RS232 je různá podle zvolené přenosové rychlosti, čím nižší je přenosová rychlost, tím větší vzdálenost lze překonat. Maximální možná vzdálenost, po které lze komunikovat popisuje tabulka 2.3, ale v dnešní době je lepší se orientovat podle údajů dodaných výrobcem zařízení nebo součástky.

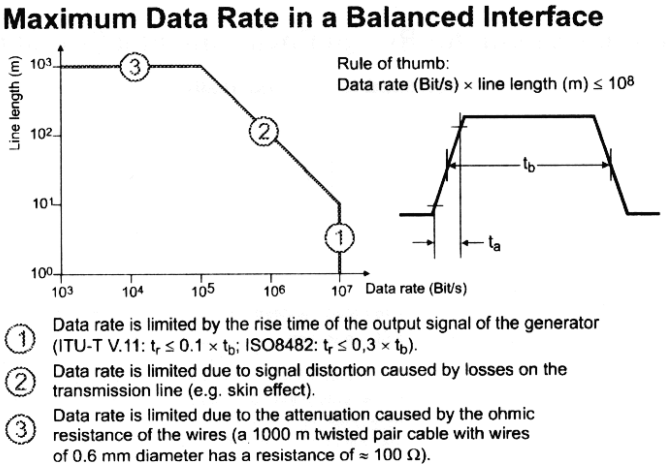
Pro přenos dat na větší vzdálenosti je výhodnější používat rozhraní RS422 nebo RS485.

Baud rate [Bd]	Maximální délka [m]
19 200	15
9 600	150
4 800	300
2 400	900

Tabulka 2.3: Maximální délka vedení RS232.

2.4 RS485

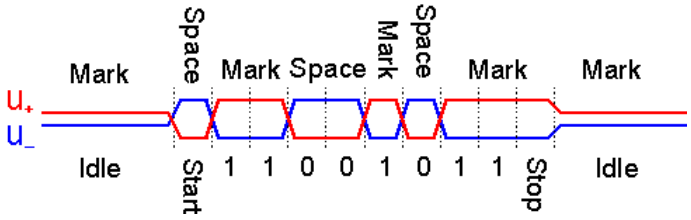
RS485 je odolná komunikační sběrnice využívaná především v průmyslových aplikacích nebo v prostředích s požadavky na vysokou odolnost proti rušení. Sběrnice vychází z rozhraní RS232, kterému byla věnována kapitola 2.3, od kterého se liší především jinými napěťovými úrovněmi, nepřítomností pomocných řídících signálů a možností multipoint komunikace. Sběrnici RS485 lze vytvořit z rozhraní RS232 pomocí převodníků úrovně. RS485 využívá pro komunikaci kroucený pár, kde jeden vodič je označován písmenem A (invertující pin, který je negativní, když je vedení nečinné) a druhý vodič písmenem B (neinvertující pin, který je pozitivní, když je vedení nečinné), vodiče někteří výrobci označují opačně a někteří také jako RxTx- a RxTx+. Maximální délka sběrnice je 1200 m a lze ji prodloužit za použití opakovačů. Přenosová rychlost je nepřímo úměrná délce vedení a u krátkých vedení lze dosáhnout rychlosti do 10 Mbit/s. Závislost přenosové rychlosti na délce vedení popisuje obrázek 2.5 (obrázek převzat z článku [4]). Standard definuje použití maximálně 32 zařízení na jedné větvi. Zařízení musí být zapojována za sebou do sběrnice, nikoliv do hvězdy. Komunikace na lince funguje v poloduplexním režimu, to znamená, že v danou chvíli může vysílat pouze jedno zařízení (systém dotaz-odpověď).



Obrázek 2.5: Závislost přenosové rychlosti na délce vedení RS485.

Stejně jako RS232 využívá RS485 dvě napěťové úrovně pro komunikaci s tím rozdílem, že logické úrovně jsou reprezentovány rozdílem napětím mezi oběma vodiči a nejsou

vztaženy k referenční zemi. Tímto systémem rozeznávání logických stavů je potlačena možnost rušení, jelikož indukovaný rušivý signál se většinou přičítá k oběma vodičům stejně. Logické stavy jsou detekovány, pokud rozdíly mezi vodiči A a B jsou větší jak 200 mV nebo menší jak -200 mV. Logické stavy jsou označovány stejně jako u RS232, logická 1 označována jako „mark“ a logická 0 jako „space“. Pro přenos informací se využívá 7 nebo 8bitový rámec se start bitem, jedním nebo zdvojeným stop bitem a volitelným paritním bitem. Podobně jako u RS232 je start bit realizován logickou 0 tedy „space“ a stop bit logickou 1 tedy „mark“. Na obrázku 2.6 (obrázek převzat z článku [2]) je vyobrazen přenos rámce RS485 s vyznačeným průběhem napěťových úrovní.



Obrázek 2.6: Rámec RS485.

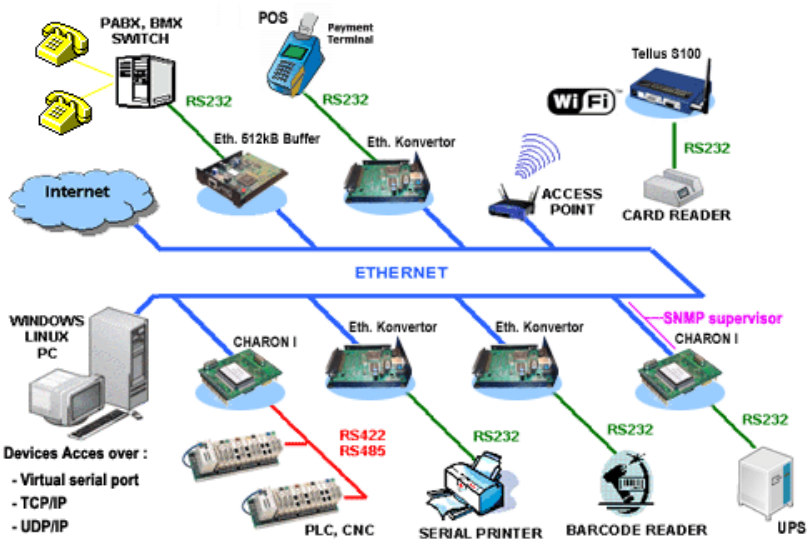
2.5 Virtuální sériový port

Virtuální sériový port je softwarový ovladač, který vytvoří v operačním systému zdánlivý sériový port a data určená pro tento port přesměruje na jiné hardwarové zařízení, nejčastěji připojené pomocí USB nebo přes počítačovou síť. Pomocí této technologie lze připojit zařízení se sériovým portem k počítači i v případě, že by připojení bylo jinak obtížně proveditelné, například z důvodu chybějícího sériového portu na novějších počítačích, velké vzdálenosti mezi zařízením a počítačem nebo ochrany počítače před elektrickým poškozením. Většina VSP přenáší pouze základní datový tok a parametry, jako rychlost přenosu v baudech, počet datových bitů, potvrzení a parita jsou pevně nastaveny na koncovém konvertoru Ethernet-sériové rozhraní. Problémem u VSP může být nezaručená doba doručení paketu na fyzické rozhraní sériového portu. U rozhraní RS232 se jedná řádově o mikrosekundy, pro USB může zpoždění dosahovat 16 ms a v případě přenosu přes počítačovou síť se doba zpoždění může pohybovat od jednotek milisekund po desítky milisekund. Pro komunikační protokoly, které využívají potvrzování velmi krátkých zpráv a mají krátký timeout, může být dlouhá doba odezvy velký problém.

Virtuální sériový port najde uplatnění zejména v případech kdy:

- není na počítači k dispozici rozhraní RS232,
- není na počítači k dispozici sběrnice RS485,
- vzdálenost mezi počítačem a zařízením s rozhraním RS232 nebo RS485 je velká,
- zařízení s rozhraním RS232 nebo RS485 má využívat více uživatelů.

Obrázek 2.7 (obrázek převzat z článku [5]) znázorňuje možnosti aplikace virtuálního sériového portu v praxi.



Obrázek 2.7: Možnosti aplikace VSP.

2.6 C#

Jedná se o vysokoúrovňový objektově orientovaný programovací jazyk založený na jazyce C++ a jazyce Java. Jazyk byl vyvinut firmou Microsoft zároveň s platformou .Net Framework a schválen standardizačními komisemi ECMA a ISO. C# lze využít k tvorbě široké škály aplikací například databázových programů, formulářových aplikací pro operační systémy Windows, webových aplikací a stránek. Stejně jako jazyky C++ a Java je C# case-sensitive (významově rozlišuje malá a velká písmena, slovo promenna není to samé co Promenna, tato slova jsou brána jako dvě rozdílná slova). V jazyce neexistují globální proměnné a metody. Požadavkem pro běh aplikací toho jazyka je prostředí .Net Framework. Další základní charakteristiky jazyka jsou:

- obsahuje nativní podporu komponentového programování,
- jednoduchá dědičnost a možnost vícenásobné implementace rozhraní,
- události,
- garbage collector se stará o uvolňování zdrojů (automatická správa paměti),
- zajišťuje typovou bezpečnost,
- podporuje zpracování chyb a výjimek.

3 Návrh a realizace zapojení

V této kapitole jsou popsány důvody a realizace galvanického oddělení měřící sestavy a dostupnosti přes Ethernetovou počítačovou síť.

3.1 Důvod požadavku dostupnosti pomocí technologie Ethernet

Požadavek na dostupnost přes počítačovou síť Ethernet byl zadán z důvodu zvýšení komfortu pracovníků, zjednodušení ovládání celé zkušební sestavy a dostupnosti sestavy, ze kteréhokoliv počítače ve firemní síti LAN. Doposud se musel pokaždé ke zkušební sestavě připojovat notebook, ze kterého se provádělo ovládaní zkušební sestavy. Pracovník tedy musel sestavit zkušební sestavu a připojit notebook, zapnout programové ovládaní a poté v průběhu testu opouštět své pracoviště, aby kontroloval stav programového ovládaní, zda nedošlo k havárii z důvodu uvedených v úvodní kapitole práce. Pokud by byla zkušební sestava dostupná přes počítačovou síť Ethernet, tak by nebylo nutné pokaždé připojovat notebook ke zkušební sestavě a pracovník by mohl kontrolovat stav programového ovládaní ze svého pracoviště.

3.2 Realizace dostupnosti pomocí technologie Ethernet

Jelikož programové ovládání využívá pro komunikaci s komponentami zkušební sestavy rozhraní RS232, tak se nabídla možnost vyřešit dostupnost zkušební sestavy přes počítačovou síť pomocí síťového serveru sériového portu a virtuálního sériového portu. Virtuální sériový port popisuje kapitola 2.5.

Kvůli požadavku na výběr komponent ze skladových zásob firmy jsem byl omezen na výrobek firmy Moxa. Vybral jsem síťový server sériového portu Moxa NPort IA-5150-S-SC, který disponuje optickým rozhraním pro připojení jednomódového optického vlákna, rozhraním RS232 a RS422/485. Na zařízení jsem nastavil parametry sítě (IP adresu, masku sítě) a parametry pro sériové rozhraní (rychlost v baudech, paritu, počet bitů, stop bit). Poté bylo nutné nainstalovat do počítače virtuální sériový port pomocí programu pro správu virtuálních portů NPort Windows Driver Manager firmy Moxa.

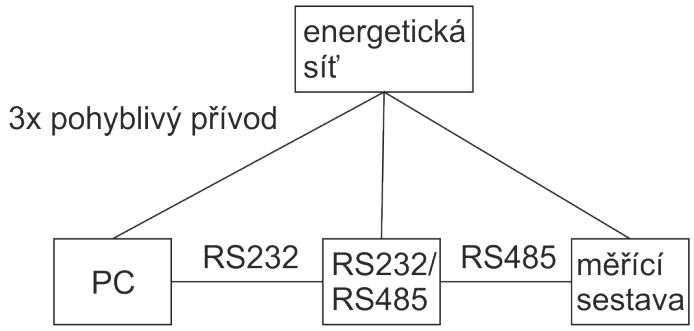
3.3 Důvod požadavku galvanického oddělení

Galvanické oddělení je způsob oddělení dvou nebo více částí obvodu tak, aby nebyly v žádném bodě vzájemně propojené, ale docházelo k přenosu dat, impulsů nebo informací. Galvanické oddělení je možné realizovat různými způsoby, například:

- transformátorová vazba (princip elekromagnetické indukce),
- optická vazba (princip elektro-optické konverze),
- mechanická vazba (princip elektromagnetu: relé, stykač).

Důvodem požadavku na galvanické oddělení měřící sestavy od řídícího počítače je možnost odrazu a naindukování vysokofrekvenční energie z vazebních členů do obvodů

nepodléhající zkoušce, jako komunikační vedení a energetické vedení. Tato energie může vyřadit nebo dokonce poškodit zařízení mimo zkušební sestavu, například řídící počítač, což není žádoucí. Tento jev se již vyskytl v minulosti, kdy se řídící počítač samovolně restartoval z důvodu této vysokofrekvenční energie. Proto byl vznesen požadavek na galvanické oddělení měřící soustavy od řídícího počítače, aby se předešlo těmto možným problémům. Na obrázku 3.1 je zobrazeno blokové schéma původního zapojení měřící soustavy.



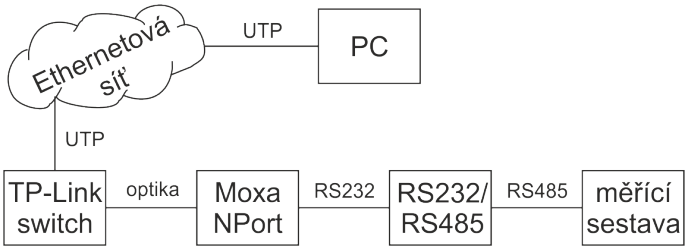
Obrázek 3.1: Blokové schéma původního zapojení měřící sestavy.

3.4 Realizace galvanického oddělení

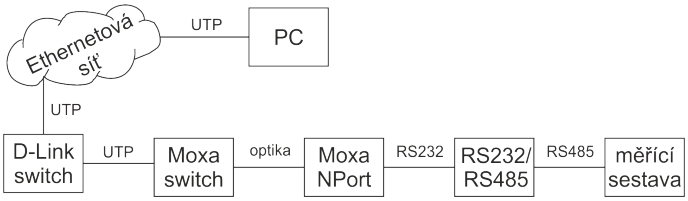
Jak je zmíněno v předchozí kapitole 3.3 existuje více možností realizace galvanického oddělení. Po úvaze a zhodnocení následujícího požadavku dostupnosti měřící soustavy přes počítačovou síť Ethernet jsem došel k závěru, že galvanické oddělení bude nejlépe a jednoduše proveditelné pomocí optické vazby, za využití konverze z metalického UTP kabelu počítačové sítě na optické vlákno (optický Ethernetový kabel). Toto galvanické oddělení lze jednoduše vytvořit pomocí vyráběných převodníků. Na výběr jsem měl převodníky firmy Moxa a Papouch, jelikož tyto výrobky byly ve firmě skladem. Blokové schéma této varianty popisuje obrázek 3.3. Druhou možností bylo využití zařízení počítačové sítě přepínače firmy TP-Link s SFP slotem osazeným modulem TL-SM311LS, tímto by odpadla nutnost zapojení samostatného převodníku z metalického krouceného páru na optiku a nahradil by se původní přepínač firmy D-Link v rozvaděči za přepínač firmy TP-Link s SFP slotem. Tato varianta je vyobrazena na obrázku 3.2. Druhá možnost se jevila samozřejmě jako výhodnější, protože by nebylo nutné využít dalšího zařízení (převodníku).

Pro otestování zapojení jsem osadil TP-Link přepínač modulem TL-SM311LS[1] pro jednomódové optické vlákno a propojil jsem přepínač s Moxa NPort pomocí jednomódového optického kabelu. Po zapojení napájení jsem očekával signalizaci úspěšné komunikace rozsvícenou LED na Moxa NPortu, ale LED se nerozsvítila. Provedl jsem tedy vyčištění konektorů optického kabelu a znovu jej zapojil. Problém přetrvával, tak jsem zkusil osadit i druhý SFP slot dalším optickým modulem a provedl jsem přepojení optického kabelu

[1]Single-mode MiniGBIC modul – optický transceiver, 1310 nm, optický kabel 9/125, 2x konektor LC.



Obrázek 3.2: Blokové schéma zapojení s TP-Link přepínačem.



Obrázek 3.3: Blokové schéma zapojení s Moxa přepínačem.

do druhého optického modulu. Osazení druhým modulem a jeho zapojení, také nevyřešilo problém a Moxa NPort nekomunikoval s TP-Link přepínačem. Poté jsem zkusil na přepínači propojit oba optické moduly, zda není chyba v modulech nebo přepínači. Po propojení modulů se rozsvítily kontrolky modulů na přepínači, takže chyba nebyla na straně modulů nebo přepínače. Ověřil jsem nastavení přepínače, zkontroloval datové listy optických modulů a Moxa NPortu, ale neobjevil jsem žádný důvod, proč by neměly zařízení mezi sebou komunikovat.

Zůstala mi tedy možnost realizovat galvanické oddělení převodníkem. Po předchozí zkušenosti s nekompatibilitou zařízení jsem zvolil Ethernetový přepínač EDS-308-S-SC firmy Moxa s optickým slotem pro jednomódové optické vlákno. Pro vyzkoušení jsem zapojil Moxa NPort a Moxa přepínač mezi sebou a očekával signalizaci komunikace rozsvícenou LED. Po chvíli se LED rozsvítila, připojil jsem notebook pomocí UTP kabelu k Moxa přepínači a přistoupil jsem na administraci Moxa NPortu přes internetový prohlížeč. Zapojení a komunikace byla tedy úspěšná. Na fotografii 3.7 lze vidět test optického propojení.

Dalším úkolem bylo vyřešení ochrany energetické sítě před možnými účinky vysokofrekvenční energie. Pro tento účel jsem využil opět skladových zásob firmy a použil jsem přepěťovou ochranu s vysokofrekvenčním filtrem. Tato ochrana zabrání šíření vysokofrekvenční složky dále do energetické sítě. Vysokofrekvenční filtr je výrobek firmy Saltek, jedná se o typ DA-275 DF 16. Je to přepěťová ochrana s vysokofrekvenčním filtrem třídy D (3. stupeň).

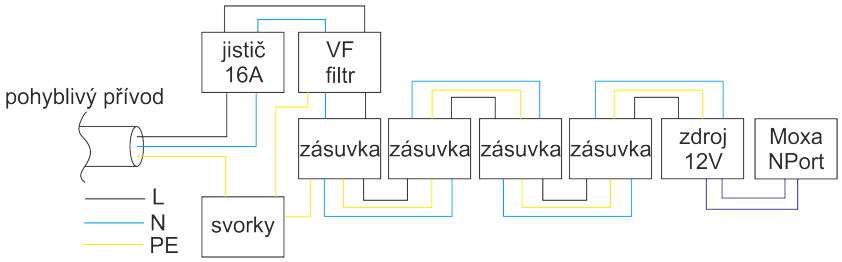
3.5 Celkové zapojení

Vzhledem k tomu, že zkušební sestava není stále sestavena na jednom místě, tak bylo potřeba komponenty nějak sestavit do jednoho celku. Sestavil jsem komponenty včetně

Moxa NPortu do plastového rozvaděče a osadil ho vývodkou, kterou je veden pohyblivý přívod. Rozvaděč je osazen komponentami:

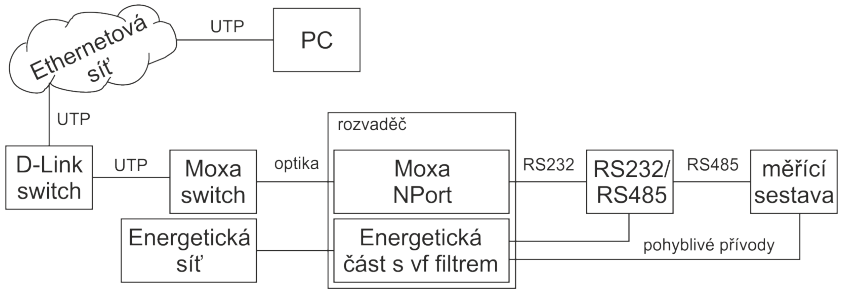
- jistič OEZ minia LPN B 10,
- přepěťová ochrana s vf filtrem Saltek DA-275 DF 16,
- zásuvky OEZ minia ZSE,
- zásuvky OEZ minia ZSF,
- zdroj DC napětí TDK-Lambda DSP 10-12,
- Moxa NPort IA-5150-S-SC.

Blokové schéma zapojení komponent v rozvaděči je znázorněno na obrázku 3.4, propojení jsem provedl lankovými vodiči průřezu 1,5 mm^2.



Obrázek 3.4: Blokové schéma zapojení rozvaděče.

Celkové zapojení galvanického oddělení a dostupnosti přes Ethernet je vyobrazeno na obrázku 3.5.



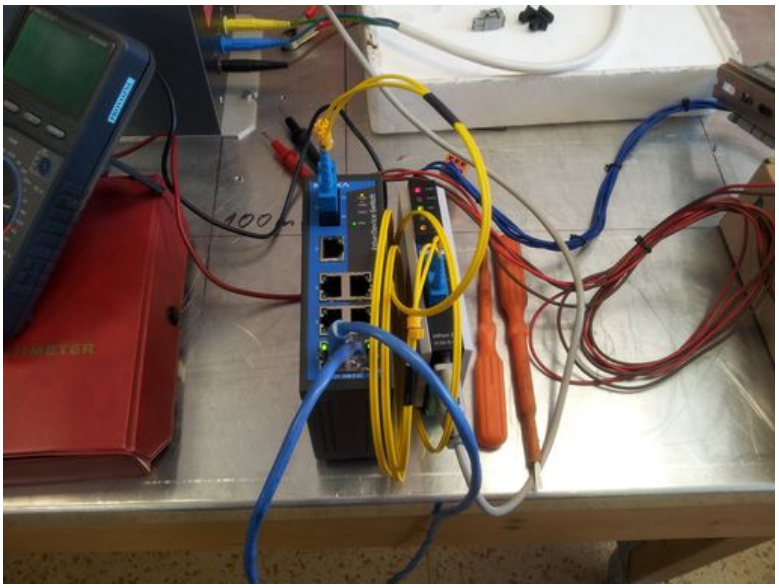
Obrázek 3.5: Blokové schéma celkového zapojení.

3.6 Ověření funkcionality zapojení

Realizované galvanické oddělení a jeho zapojení jsem následně prakticky ověřil zapojením všech navržených částí. Poté jsem připojil notebook do zásuvky sítě LAN v místnosti

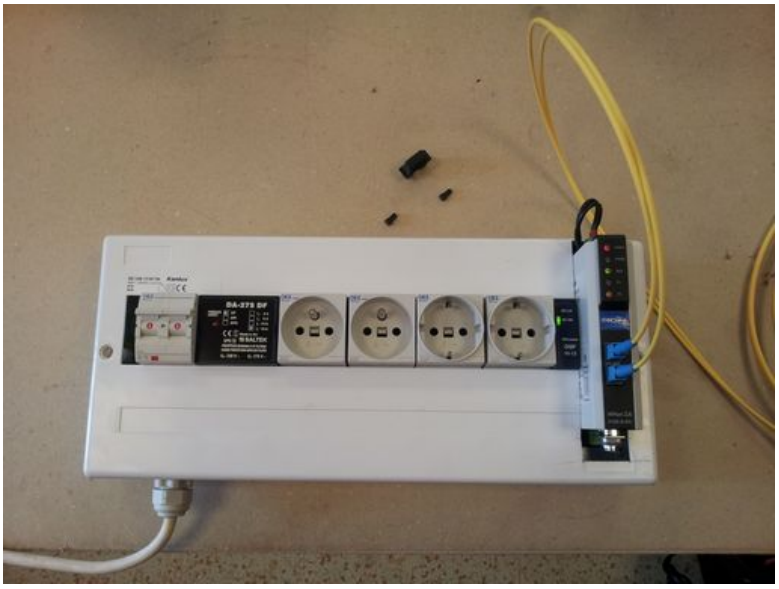
a vyzkoušel dostupnost Moxa NPortu, který byl bez problémů dostupný. Připojil jsem na Moxa NPort loopback konektor[2] a vyslal jsem na sériový port data, která se úspěšně vrátila. Na následujících fotografiích je vyobrazena původní zkušební sestava 3.6, osazený rozvaděč 3.8 a umístění Moxa přepínače 3.9.

Obrázek 3.6: Fotografie zkušební sestavy s počítačem.

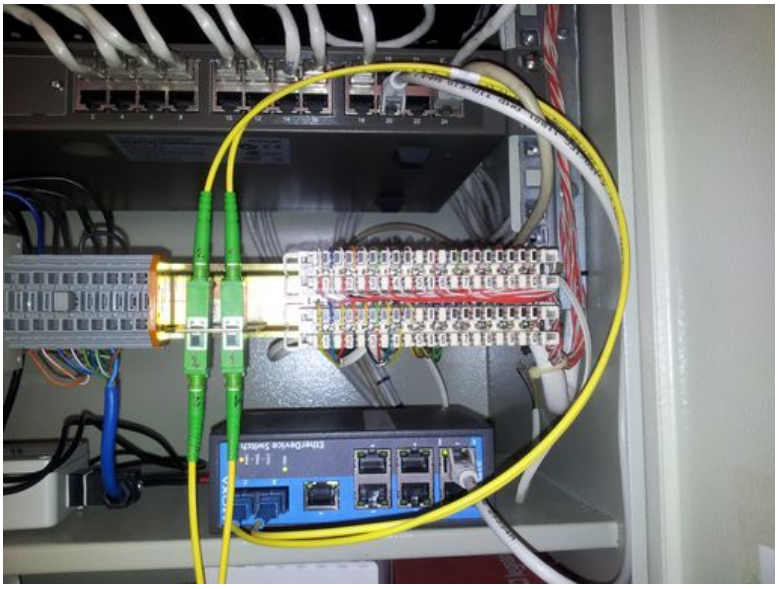


Obrázek 3.7: Fotografie testu optického spojení.

[2]Konektor zapojen jako smyčka pro testování sériového portu, kde vyslaná data jsou zaslána ihned zpět.



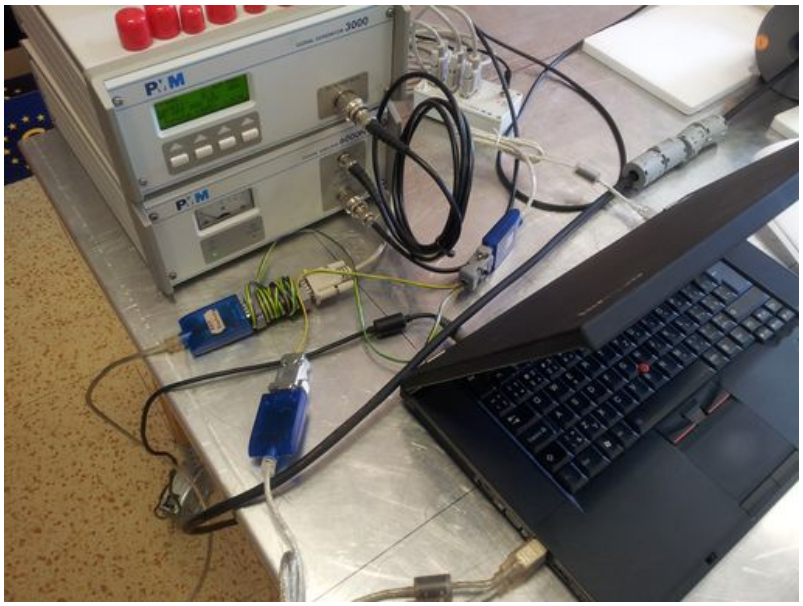
Obrázek 3.8: Fotografie vyrobeného rozvaděče.



Obrázek 3.9: Fotografie umístění Moxa přepínače v rozvaděči.

4 Generátory

Následující dvě kapitoly popisují parametry komponent ovládaných přes sériovou linku pomocí programového ovládání a popis jejich komunikace. Kapitoly obsahují pouze popis parametrů a komunikace, protože výrobce nijak nepopisuje vnitřní obvody komponent. Komunikaci jsem si kromě nastudování z manuálů komponent, také odposlechl přímo na sériové lince, obrázek 4.1. Pro připojení na sériový port jsem využil aplikaci Terminal [7]. Pomocí této aplikace jsem uložil komunikaci do textových souborů a využil ji dále v práci pro ladění programového ovládání.



Obrázek 4.1: Fotografie odposlechu komunikace na sériové lince.

4.1 PMM3000

PMM 3000 je vysokofrekvenční signální generátor s rozsahem od 10 kHz do 1000 MHz. Generátor je plně automatický a programovatelný přes sériový port RS232/485. Díky svému vybavení je ideálním nástrojem pro zkoušky odolnosti EMC. Generátor má několik funkcí, které nejsou zahrnuty v běžných generátorech, jako interní pulsní modulace a rozmítání. Podle ovládacího panelu lze na první pohled usoudit, že generátor je určen převážně pro ovládání přes sériový port. Tabulka 4.1 popisuje parametry generátoru.

Frekvence	
Rozsah	10 kHz až 1000 MHz
Rozlišení	1 kHz (při f $<$ 100 MHz), 10 kHz (při f $>$ 100 MHz)
Přesnost	$\pm$ 50 ppm
Level	
Rozsah	-80 až +10 dBm
Rozlišení	0,1 dB
Přesnost	$\pm$ 1 dB
Výstupní impedance	50 Ω
Konektor	N (zásuvka)
AM modulace	
Interní	400 nebo 1 kHz (hloubka 80 %)
Externí	100 Hz až 10 kHz (max. hloubka 90 %)
Vstupní impedance	$>$ 1 kΩ
Konektor	BNC (zásuvka)
Pulsní modulace	
Interní	200 Hz
Vyp./Zap. poměr	při 100 MHz $>$ 70 dB, při 900 MHz $>$ 40 dB
Frekvenční přesnost	$\pm$ 10 %
Pulsní modulace 1 Hz	
Interní	1 Hz
Vyp./Zap. poměr	při 900 MHz, 0 dBm $>$ 70 dB
Frekvenční přesnost	$\pm$ 10 %

Tabulka 4.1: Parametry generátoru PMM 3000

Tabulka 4.2 popisuje komunikaci mezi generátorem a programovým ovládáním. Komunikace probíhá rychlostí 9600 Bd, 8bitovou délkou, jedním stop bitem a žádnou paritou. Každý povel pro generátor a jeho odpověď je uzavřena mezi znaky # a *. Pokud je generátor ovládán přes rozhraní RS485, tak musí povel pro generátor obsahovat adresu generátoru, která následuje hned za znakem #. Adresa se nastavuje manuálně na generátoru a může být z rozsahu 01 až 99. V případě komunikace přes rozhraní RS232 je adresa ignorována. Délka odpovědi mezi hraničními znaky může být 1 až 100 znaků.

Příkaz	Odpověď	Popis
#00v*	#PMM 3000 verze datum*	dotaz na typ a verzi firmware generátoru
#00F*frekvence**		nastavení frekvence v MHz
#00?f*	#F:*frekvence*MHz*	dotaz na aktuálně nastavenou frekvenci generátoru
#00M4*		nastavení 400 Hz modulace
#00M1*		nastavení 1 kHz modulace
#00Me*		nastavení externí modulace
#00Mp*		nastavení pulsní modulace
#00M5*		nastavení 1 Hz pulsní modulace
#00M0*		vypnutí modulace
#00?m*	#MOD:*x**	dotaz na aktuálně nastavenou modulaci, *x* – znak nabývá hodnot znaku za M z předchozích příkazů nastavení modulace
#00L*level**		nastavení level v dBm
#00?l*	#Lev:*level*dBm*	dotaz na aktuálně nastavený level
#00R0*		vypnutí RF výstupu
#00R1*		zapnutí RF výstupu
#00?r*	#RF:ON* nebo #RF:OFF*	dotaz na stav RF výstupu generátoru

Tabulka 4.2: Komunikace PMM 3000

Pokud není u příkazu v tabulce zaznačena odpověď, tak je odpověď stejná jako příkaz.

4.2 SG2000

SG 2000 je vysokofrekvenční generátor s rozsahem od 100 kHz do 1000 MHz s možností amplitudové modulace, kmitočtové modulace a rozmítání. Výstupní úroveň lze nastavit v rozsahu -127 dBm až +10 dBm. Generátor je založen na principu směšování signálů ze dvou oscilátorů. Řízení zajišťují dva mikroprocesory. Pomocí sériového portu RS232 lze generátor dálkově ovládat, a tak může být uplatněn v automatizovaných měřících systémech. Přístroj je koncipován tak, aby našel uplatnění při běžných vf měřeních, výrobě a servisu. Tento generátor slouží jako doplňkový při zkoušce odolnosti EMC. Tabulka 4.3 popisuje parametry generátoru.

Komunikace probíhá rychlostí 9600 Bd, 8bitovou délkou, s jedním stop bitem a žádnou paritou. Každý povel je ukončen znakem ENTER (CR). Povely jsou potvrzeny generátorem zpětným vysláním OK nebo v případě chyby vysláním ERROR. Generátor lze ovládat pouze přes rozhraní RS232. Tabulka 4.4 popisuje komunikaci s generátorem SG 2000.

Pokud není u příkazu v tabulce zaznačena odpověď, tak je odpověď dle stavu vykonání příkazu, a to buď SOK pro úspěšné vykonání, nebo SER pro neúspěšné vykonání příkazu.

Frekvence	
Rozsah	100 kHz až 1000 MHz
Minimální kmitočtový krok	1 Hz
Přesnost	$\pm$ 1 ppm
Level	
Rozsah	-127 až +10 dBm
Rozlišení	1 dB
Přesnost	$\pm$ 0,7 dB pro f $<$ 500 MHz, $\pm$ 1 dB pro f $>$ 500 MHz
AM modulace	
Hloubka	0 až 100 %
Krok	0,1 %
Harmonické zkreslení	$<$ 3 % do hloubky 90 %
FM modulace	
Zdvih	0 až 100 kHz
Krok	0,1 kHz
Harmonické zkreslení	$<$ 0,2 %

Tabulka 4.3: Parametry generátoru SG2000

Příkaz	**Odpověď**	**Popis**
SFR*frekvence*		nastavení frekvence v Hz (desetimístné číslo)
RFR	SFR*frekvence*	dotaz na aktuálně nastavenou frekvenci generátoru v Hz
SLV*level3*		nastavení level (třímístné číslo 0 až 137)
RLV	SLV*level3*	dotaz na aktuálně nastavený level
SMD*hloubka*		nastavení AM modulace a hloubky (čtyřmístné číslo 0 až 100)
RMD	SMD*hloubka*	dotaz na aktuálně nastavenou hloubku modulace
SMS*zdvih*		nastavení FM modulace a zdvihu (čtyřmístné číslo 0 až 100)
RMS	SMS*zdvih*	dotaz na aktuálně nastavený zdvih modulace
SMI*0-2*		nastavení zdroje modulace (0 – interní, 1 – externí, 2 – sweep)
RMI	SMI*0-2*	dotaz na nastavení zdroje modulace
SMO		nastavení modulace na vypnuto
RMM	SMM*0-4*	dotaz na nastavenou modulaci (0 – AM, 1 – FM, 2 – AM OFF, 3 – FM OFF, 4 – Error

Tabulka 4.4: Komunikace SG 2000

5 Měřič výkonu

Tato kapitola se zabývá měřičem vysokofrekvenčního výkonu využitého v měřící soustavě.

5.1 PMM 6600

PMM 6600 je jednoduchý a snadno použitelný měřič vysokofrekvenčního výkonu. Měřič byl navrhnut především pro aplikace zkoušek kompatibility EMC nebo měření vstupního výkonu antén. Měřič je ovladatelný pouze přes rozhraní RS485 a nedisponuje žádnou zobrazovací jednotkou. Napájení je zajištěno pomocí dalšího vodiče v kabelu s linkou RS485, ze kterého je měřič napájen $\pm$ 12 V. Parametry měřiče jsou popsány v tabulce 5.1.

Frekvence	
Rozsah	10 kHz až 1000 MHz
Výkon	
Rozsah	10 nW až 0,5 W (-40 – +27 dBm)
Maximální výkon	1W
Chyba měření	
10 kHz – 100 kHz	$\pm$ 1,5 dB
100 kHz – 1000 MHz	$\pm$ 1 dB
Obecné parametry	
Napájení	$\pm$ 12 V DC z RS485
Konektor	N (zásuvka)
Vstupní impedance	50Ω
SWR	1,25

Tabulka 5.1: Parametry vf měřiče PMM6600

Komunikace s měřičem je asynchronní, probíhá rychlostí 9600 Bd, s 8bitovou délkou, jedním stop bitem a žádnou paritou. Tabulka 5.2 popisuje komunikaci s měřičem.

Příkaz	**Odpověď**	**Popis**
#PMv*	PMM6600 Vx	dotaz na typ a verzi firmware měřiče
#PMp*	*HL*	dotaz na aktuálně naměřený výkon (odpověď je byte, výpočet výkonu se provede podle vzorce 5.1)

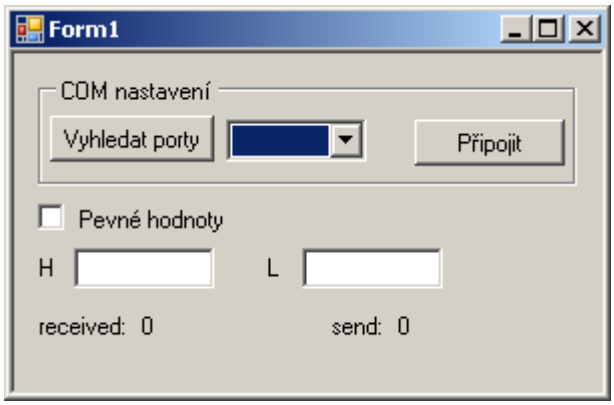
Tabulka 5.2: Komunikace PMM 6600

Výpočet naměřeného výkonu se provádí z přijatých hodnot formátu bajt podle vzorce 5.1. Výsledkem výpočtu je hodnota výkonu v dBm.

$$P = \frac{(H * 256 + L) - 1000}{10} \tag{5.1}$$

6 Návrh a realizace uživatelského rozhraní

Programové ovládání je rozděleno do tří samostatných aplikací podle jejich primárního určení a reflektuje požadavky na jejich funkcionalitu. Protože tato práce byla mým prvním seznámením s obsluhou sériového portu, tak jsem se rozhodl začít s jednodušší části programového ovládaní, a to s programem pro měřič výkonu. Pro nastudovaní a praktické vyzkoušení obsluhy sériového portu jsem si naprogramoval malou aplikaci simulující měřič výkonu, která odpovídá na příkazy přijaté na sériovém portu. Rozhraní aplikace je vyobrazeno na obrázku 6.1. Aplikaci jsem odladil pomocí programu Terminal [7], a tak jsem měl připravenou plnohodnotnou náhradu za měřič výkonu.



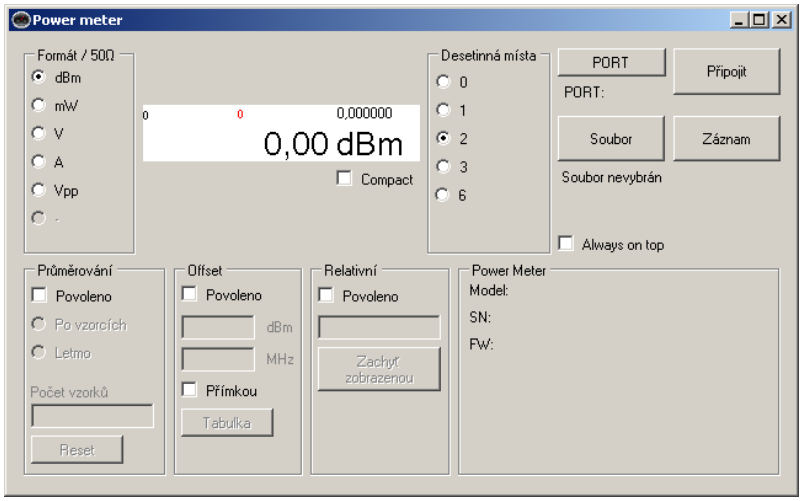
Obrázek 6.1: Obrázek rozhraní simulátoru měřiče výkonu.

6.1 Uživatelské rozhraní měřiče výkonu

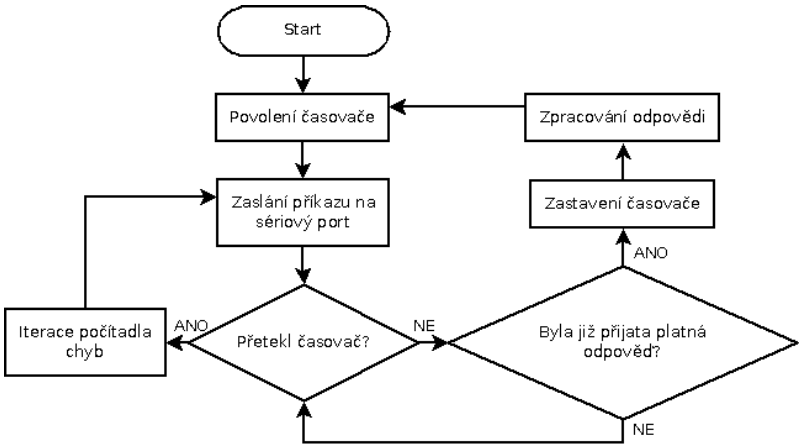
Rozhraní měřiče výkonu mělo být rozšířeno proti původnímu rozhraní o možnosti:

- zobrazení naměřeného výkonu v určených veličinách,
- průměrování naměřených hodnot,
- offset z tabulky,
- relativní hodnoty offsetu,
- volby počtu desetinných míst,
- záznamu do souboru ve formátu csv.

Rozložení komponent v rozhraní nebylo omezeno žádnou podmínkou nebo požadavkem. Obrázek 6.2 zobrazuje uživatelské rozhraní měřiče výkonu. Vyčítání hodnot z měřiče je závislé především na rychlosti linky a hardwaru, poněvadž dotazování na naměřenou hodnotu měřiče je zkonstruováno tak, že je inicializován první dotaz na typ a verzi měřiče výkonu, jakmile je přijata odpověď ve správném tvaru, tak se odešle další příkaz na



Obrázek 6.2: Obrázek rozhraní měřiče výkonu.



Obrázek 6.3: Vývojový diagram vyčítání dat z měřiče výkonu.

sériový port. Tento proces znázorňuje vývojový diagram 6.3. Vyčítání hodnot je zastaveno v okamžiku odpojení od sériového portu.

Pro získání dostupných sériových portů jsem si napsal metodu uvedenou ve výpisu 1, která vrací všechny aktuálně dostupné sériové porty v počítači.

```
private List<string> ScanAvailAblePorts()
{
    List<string> ports = new List<string>(SerialPort.GetPortNames());
    ports.Sort();
    if (ports.Count == 0) MessageBox.Show("Zadny COM port neni dostupny.");
    return ports;
}
```

Výpis 1: Metoda vracející dostupné sériové porty v počítači.

Při vyčítání dat ze sériového portu se stávalo, že očekávaná data nedorazila všechna ihned, ale dorazila až s odpovědí na následující příkaz. Tento jev je zapříčiněný různými buffery (v převodníku, operačním systému). Pro úplné vyčtení dat jsem využil kód ve výpisu 2, která načítá data ze sériového portu do lokálního bufferu, se kterým se dále pracuje.

```
if (serialPort1.IsOpen)
{
   while (serialPort1.BytesToRead > 0)
   {
      Buffer.Add((byte)serialPort1.ReadByte());
   }
}
```

Výpis 2: Načítání dat do lokálního bufferu.

Po načtení dat do lokálního bufferu se přijatá data testují, zda obsahují předpokládané odpovědi. Pokud mají data v bufferu dostatečnou délku a zároveň je nastaven příznak pro očekávání odpovědi s informací o typu a verzi měřiče, tak se otestuje aktuální pozice bufferu a dvě následující na výskyt znaků uvozující začátek odpovědi. Poté se testují následující pozice bufferu, zda obsahují znak ukončující odpověď. Pokud je tento znak nalezen, tak je odpověď vykopírovaná z bufferu do lokální proměnné a dále zpracována.

Obdobný postup je aplikován i pro rozeznání odpovědí s hodnotou naměřeného výkonu. Pokud mají data v bufferu dostatečnou délku a na pozici je číslo s hodnotou 0 až 4, tak se otestuje, zda je na následující pozici číslo s hodnotou 0 až 255. Pokud se podmínky vyhodnotí jako pravdivé, tak se data z bufferu zpracují. Oba postupy jsou zachyceny ve výpisu 3.

Při zpracováváni informací z přijatých dat se vyskytl problém v okamžiku, kdy jsem chtěl zobrazit přijatá data v uživatelském rozhraní. Událost zajišťující příjem dat není zpracovávána ve stejném vlákně a není tedy součástí hlavního vlákna, ve kterém je zpracováváno uživatelské rozhraní. Tento problém jsem vyřešil pomocí delegáta, kterého jsem spustil pomocí BeginInvoke z vlákna uživatelského rozhraní. Volání delegáta je zachyceno ve výpisu 3.

```
int BufferCount = Buffer.Count;
for (int i = 0; i < BufferCount; i++)
{
    if (Buffer.Count > (i+2) && expectingVersionInfo)
    {
        if ((Buffer[i] == 'P') && (Buffer[i + 1] == 'M') && (Buffer[i + 2] == 'M'))
        {
            for (int y = i + 2; y < BufferCount; y++)
            {
                if (Buffer[y] == '\0')
                {
                    int size = y - i;
                List<byte> pom = Buffer.GetRange(i, size + 1);
                string command = string.Empty;
                for (int x = 0; x < (pom.Count - 1); x++)
                {
```

```
                    command += Convert.ToChar(pom[x]);
                }
                Buffer.RemoveRange(0, y);
                BufferCount -= y;
                timerTimeout.Enabled = false;
                this.BeginInvoke(new BufferCallback(ProcessingBuffer), new object[] { command
                    });
                break;
                 }
            }
        }
    }
    else if (Buffer.Count > i + 1)
    {
        if (Buffer[i] >= 0 && Buffer[i] < 5)
        {
            if (Buffer[i + 1] >= 0 && Buffer[i + 1] <= 255)
            {
                byte []command = {Buffer[i],Buffer[i + 1]};
                timerTimeout.Enabled = false;
                this.BeginInvoke(new BufferCallbackValues(ProcessingBuffer), new object[] {
                    command });
                Buffer.RemoveRange(0, i + 2);
                BufferCount -= i + 2;
            }
      }
    }
}
```

Výpis 3: Rozeznání odpovědí v přijatých datech

Výpočet naměřeného výkonu z přijatých dat je vypočítán pomocí vzorce 5.1 v kapitole 5.1.

Jedním z požadavků na úpravu uživatelského rozhraní byla možnost zobrazení naměřených hodnot v jednotkách:

- decibely,
- watty,
- volty,
- ampéry,
- volty špička – špička.

Jelikož měřič výkonu dodává údaje pro výpočet výkonu v decibelech, tak se musí pro zobrazení hodnoty výkonu v jiných jednotkách provádět převod. Převod je uskutečňován podle vzorců:

- 6.1 pro výsledek ve wattech,
- 6.2 pro výsledek ve voltech,

- 6.3 pro výsledek ve ampérech,
- 6.4 pro výsledek ve voltech špička – špička.

$$P = 10^{\frac{x}{10}} \tag{6.1}$$

$$U = \sqrt{\frac{10^{\frac{x}{10}}}{1000} * 50} \tag{6.2}$$

$$I = \sqrt{\frac{\frac{10^{\frac{x}{10}}}{1000}}{50}} \tag{6.3}$$

$$U_{pp} = \sqrt{\left(\frac{10^{\frac{x}{10}}}{1000} * 50\right)} * \sqrt{2} * 2 \tag{6.4}$$

Průměrování naměřených hodnot lze provádět dvěma způsoby, a to letmo nebo po vzorcích. Při průměrování letmo se každý nově naměřený vzorek zařadí do fronty, nejstarší vzorek se vyřadí z fronty a vypočítá se průměrná hodnota celé fronty. Dochází tedy k pozvolnému průběhu průměrování. Průměrování po vzorcích naplní celou frontu vzorky, poté provede výpočet průměru celé fronty a frontu vyprázdní pro nové vzorky. Algoritmus průměrování je zobrazen ve výpisu 4.

```
if (prumerovaniEnabled && checkPrumerovaniInput())
{
    int numberOfSamples = Convert.ToInt32(textBoxPocetVzorku.Text);
    if (samplesList == null) samplesList = new List<double>(numberOfSamples);
    if (samplesList.Count < numberOfSamples)
    {
        samplesList.Add(measuredValue);
    }
    else if (radioButtonPoVzorcich.Checked)
    {
        finalValue = calculateAvarageValue(samplesList);
        prumerovaniPrint = true;
        samplesList.Clear();
        samplesList.Add(measuredValue);
    }
    else if (radioButtonLetmo.Checked)
    {
        finalValue = calculateAvarageValue(samplesList);
        prumerovaniPrint = true;
        samplesList.RemoveAt(0);
        samplesList.Add(measuredValue);
    }
}
```

Výpis 4: Průměrování naměřených vzorků

Funkcí offsetu je možnost přičíst určitou hodnotu k hodnotě výkonu. Přičítaná hodnota je získaná z tabulky, kde pro každou hodnotu frekvence je přiřazena hodnota výkonu. Výběr hodnoty výkonu z tabulky je možno provádět dvěma způsoby, a to kdy je k zadané hodnotě frekvence nalezena nejbližší frekvence z tabulky nebo pomocí přímky. Pokud má být offset získán přímkou, tak se hodnota offsetu vypočítá pomocí vzorce 6.5, kde x_1 je nejbližší nižší hodnota frekvence v tabulce, x_2 nejbližší vyšší frekvence v tabulce, y_1 je hodnota výkonu pro frekvenci x_1, y_2 je hodnota výkonu pro frekvenci x_2, x je hodnota zadané frekvence a y je výsledná hodnota výkonu (offsetu).

$$\begin{aligned} N &= \frac{y_2 - y_1}{x_2 - x_1} \\ K &= y_1 - x_1 * N \\ y &= x * N + K \end{aligned} \tag{6.5}$$

Tabulka pro výběr offsetu je ve formátu xml se strukturou vyobrazenou ve výpisu 5.

```
<offsetTable>
   <item id="1">
      <mhz>1</mhz>
      <dbm>1</dbm>
   </item>
</offsetTable>
```

Výpis 5: Ukázka tabulky pro offset

Výpočet offsetu je prováděn v programu pomocí funkce naznačené ve výpisu 6.

```
public offset findOffset(double mhz, bool primkou)
{
    offset nearest = new offset("0","0");
    bool notBetween = false;
    if (!primkou)
    {
        foreach (offset x in p_offsetList)
        {
            if ((mhz - x.MHz) >= 0) nearest = x;
            if ((mhz - x.MHz) == 0) break;
        }
        return nearest;
    }
    else
    {
        offset floor = new offset(); ;
        offset ceiling = new offset();
        foreach (offset x in p_offsetList)
        {
            if ((mhz - x.MHz) >= 0) floor = x;
            if ((mhz - x.MHz) == 0)
            {
                notBetween = true;
                break;
            }
        }
```

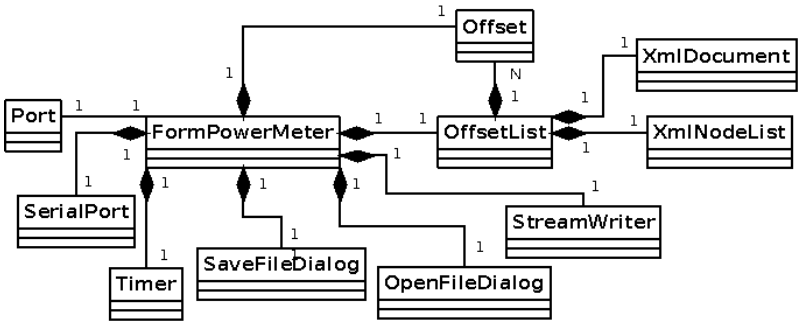
```
        if (!notBetween)
          {
              offset pom = new offset();
              bool first = true;
              foreach (offset x in p_offsetList )
              {
                  if ((mhz − x.MHz) < 0)
                  {
                      if ( first )
                      {
                          pom = x;
                          ceiling = x;
                          first = false;
                      }
                      else
                      {
                          pom = x;
                      }
                      if (pom.MHz < ceiling.MHz) ceiling = pom;
                  }
              }
              double x1 = floor.MHz;
              double y1 = floor.dBm;
              double x2 = ceiling.MHz;
              double y2 = ceiling.dBm;
              double n = ((y2 − y1) / (x2 − x1));
              double k = (y1 − (x1 ∗ n));
              double y = (mhz ∗ n) + k;
              nearest = new offset(mhz, y);
          }
          else
          {
              nearest = floor ;
          }

          return nearest;
      }
  }
```

Výpis 6: Algoritmus výpočtu offsetu

Funkce relativní hodnoty je podobná funkci offsetu, ale s tím rozdílem, že se k hodnotě naměřeného výkonu přičítá pevně zadaná hodnota v dBm.

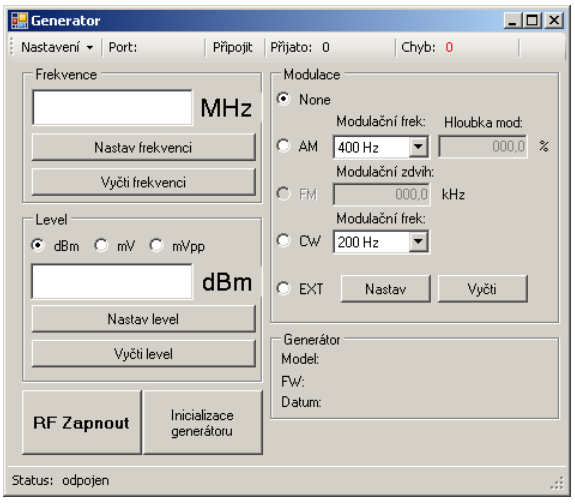
Rozložení tříd v programu znázorňuje třídní diagram 6.4.



Obrázek 6.4: Třídní diagram programu měřiče výkonu.

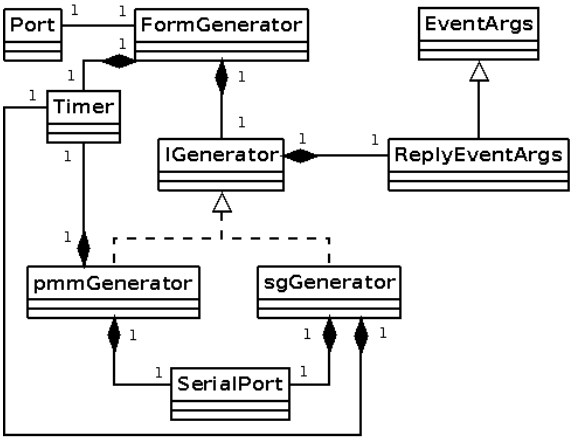
6.2 Uživatelské rozhraní generátoru

Původní programové ovládání neumožňovalo ovládání generátorů jiných výrobců a nebylo plnohodnotným nahrazením ovládacího panelu generátoru, protože programové ovládání bylo určeno výhradně pro zkoušku kompatibility EMC. Proto vznikl požadavek vyvinout plnohodnotnou náhradu za ovládací panel generátoru. Dalším požadavkem byla možnost v budoucnu doiplementovat řízení dalšího generátoru. Tak jako u rozhraní pro měřič výkonu, ani zde nebylo žádné omezení na rozložení uživatelského rozhraní. Výsledné uživatelské rozhraní je vyobrazeno na obrázku 6.5, kde pod položkou *Nastavení* jsou možnosti výběru sériového portu, výběru typu generátoru a nastavení adresy RS485 pro generátor PMM 3000.



Obrázek 6.5: Uživatelské rozhraní generátoru.

Aby bylo možné ovládat různé druhy generátorů a případně doiplementovat řízení dalších generátorů, tak jsem využil rozhraní, které implementují jednotlivé třídy generátorů a zformoval odpovědi tříd generátorů na zaslané příkazy do zpráv stejného formátů. Pro sjednocení odpovědí jsem využil událostí, u kterých jsem si vytvořil vlastní argumenty. Každá třída generátoru obsluhuje svou třídu sériového portu. Detekce přijatých zpráv je založena na kódu ve výpisu 3 v kapitole 6.1 s rozdílem, že počáteční a koncové znaky uvozující zprávu jsou jiné. Třídy programu zobrazuje třídní diagram 6.6.



Obrázek 6.6: Třídní diagram programu generátoru.

Odpovědi tříd jsou ve formátu, kde prvním parametrem je kód odpovědi (například zda komunikace běží, úspěšně proběhla, naskytla se chyba) a druhým parametrem je obsah odpovědi (například hodnota dotazovaného parametru). Odpověď se poté zpracovává ve třídě formuláře. Při zpracovávání odpovědí jsem narazil na stejný problém jako u práce se sériovým portem u uživatelského rozhraní pro měřič výkonu. Problém byl opět v komunikaci mezi vlákny. Tento problém jsem vyřešil pomocí kontroly, zda je nutné zavolat metodu ze stejného vlákna, a pokud je podmínka vyhodnocena jako pravdivá, tak se pomocí Invoke a MethodInvoker provede následující kód ve stejném vlákně, ve kterém běží uživatelské rozhraní. Pro lepší pochopení je algoritmus zobrazen ve výpisu 7.

```
private void replyReceived(object s, ReplyEventArgs e)
{
    if (this.InvokeRequired)
    {
        Invoke(new MethodInvoker(() =>
        {
            // priklad implementace
            if (e.Code == -1) toolStripStatusLabel2.Text = "TimeOut";
            if (e.Code == 0 && e.Value.Contains("read"))
            {
                toolStripStatusLabel2.Text = "Pripraven";
                printValues(e.Value);
            }
        }));
```

```
        }
        else
        {
                // priklad implementace
                if (e.Code == -1) toolStripStatusLabel2.Text = "TimeOut";
                if (e.Code == 0 && e.Value.Contains("read"))
                {
                    toolStripStatusLabel2.Text = "Pripraven";
                    printValues(e.Value);
                }
        }
    }
```

Výpis 7: Algoritmus vyřešení komunikace mezi vlákny.

Pomocí rozhraní jsem vytvořil sjednocené volání funkcí nastavování a vyčítání parametrů generátorů. V programu poté podle podmínky vytvářím instanci jednoho typu generátoru nebo druhého typu. Rozhraní mi poté umožňuje volat metody se stejným významem jedné instance pro jakékoliv typy generátorů. Ve výpisu 8 je znázorněno větvení dle vybraného typu generátoru a nastavení frekvence generátoru.

```
// tridni atribut
IGenerator generator;

// vetveni dle vybraneho generatoru
if (selectedGenerator == EGenerator.SG2000)
{
    generator = new sgGenerator();
    generator.replyEvent += new EventHandler<ReplyEventArgs>(replyReceived);
    generator.countUpdateEvent += new EventHandler<ReplyEventArgs>(updateErrRcvCount);
}
else
{
    generator = new pmmGenerator(address);
    generator.replyEvent += new EventHandler<ReplyEventArgs>(replyReceived);
    generator.countUpdateEvent += new EventHandler<ReplyEventArgs>(updateErrRcvCount);
}

// ukazka volani nastaveni frekvence
generator.setFrequency(12);
```

Výpis 8: Inicializace vybraného generátoru.

Dále jsem musel nalézt řešení pro případ, kdy se má odeslat více příkazů na sériový port za sebou, protože by se následující příkaz mohl odeslat ještě v době, kdy neskončila předchozí komunikace a neznali bychom výsledek předchozího příkazu. Tento případ by šel vyřešit vložením příkazů pro uspání vlákna mezi odesílané příkazy, ovšem toto není z mého pohledu dobré řešení a mohlo by nést více chyb. Sestavil jsem si tedy metodu, která pozastaví vykonávání příkazů na dobu nezbytně nutnou. Metoda pracuje s časovačem, který funguje jako timeout časovač a pokud přeteče v průběhu cyklu (zajišťuje pozastavení vykonávání příkazů), cyklus je tímto ukončen. Časovač slouží jako ochrana, aby cyklus v případě chyby neiteroval do nekonečna. Cyklus je také ukončen proměnnou, která je

v každé iteraci aktualizovaná a obsahuje informaci o stavu komunikace. V cyklu se provádí překreslování uživatelského rozhraní a metoda pro zpracování událostí. Celá metoda je uvedena ve výpisu 9.

```
private void waitForReplyOfPreviousCommand()
{
    timeoutTimer.Enabled = true;
    timeout = false;
    int status = generator.Status;
    while (!timeout && (status != 0) && !(status < 0))
    {
        status = generator.Status;
        this.Refresh();
        Application.DoEvents();
    }
    if (generator.Status == -1) timeout = true;
    timeoutTimer.Enabled = false;
}
```

Výpis 9: Prodleva pro vykonání předchozího příkazu.

Po připojení programu k sériovému portu jsou z generátoru automaticky vyčteny informace o nastavených parametrech. Pro nastavení všech parametrů generátoru slouží tlačítko *Inicializace generátoru*, které po stisknutí postupně odešle všechny parametry. Tlačítko RF zapnout slouží pouze pro generátor PMM 3000, u kterého lze zapínat a vypínat výstupní obvod.

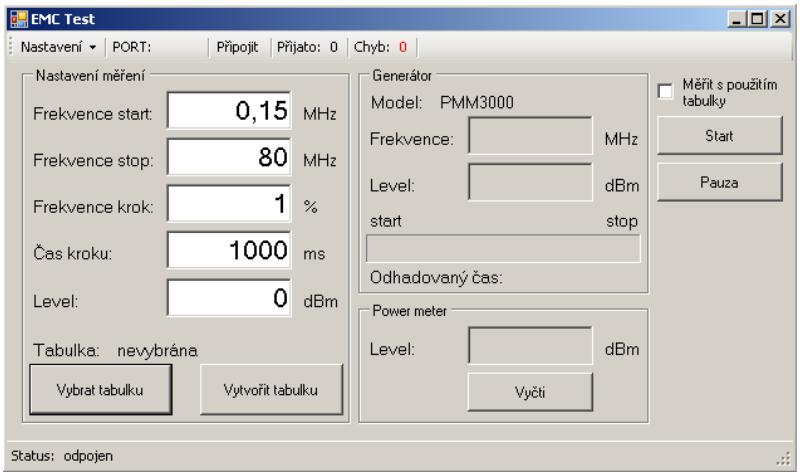
Hodnotu úrovně (level) je možné nastavit v jednotkách decibel, volt a volt špička-špička, přičemž samotné nastavení na generátoru probíhá v decibelech. Pro převod mezi jednotkami se využívá vzorců 6.1, 6.2 a 6.4 v kapitole 6.1.

6.3 Uživatelské rozhraní zkušební aplikace

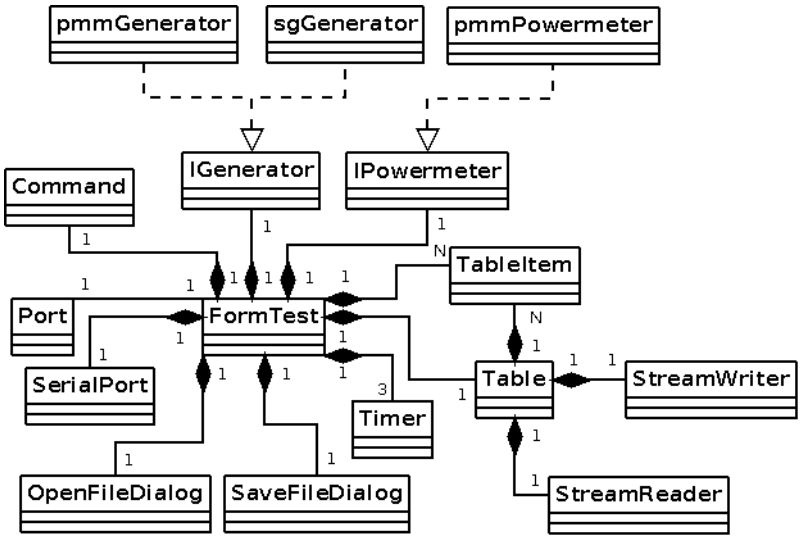
Uživatelské rozhraní obsluhující generátor a měřič výkonu v průběhu zkoušky kompatibility EMC, byl hlavní důvod pro nový vývoj. Během psaní posledního programu jsem zužitkoval nabyté poznatky během vývoje předchozích aplikací. Rozhraní je vyobrazeno na obrázku 6.7. Pod položkou *Nastavení* jsou možnosti výběru sériového portu, výběru typu generátoru a nastavení adresy RS485 pro generátor PMM 3000. V programu lze načíst tabulku ve formátu csv, tato tabulka slouží jako zdroj informací pro nastavování hodnoty úrovně (level) a následné uložení naměřené hodnoty úrovně. Pokud není tabulka načtena, není povolena volba měření s využitím tabulky, nebo není v tabulce nalezena hodnota úrovně k příslušné hodnotě frekvence, použije se pro nastavení generátoru hodnota úrovně nastavená ve formuláři. Během měření je zobrazována hodnota aktuálně nastavené frekvence, hodnoty úrovně a naměřené hodnoty úrovně. Pro lepší přehled je průběh měření vizualizován ukazatelem průběhu a výpočtem odhadovaného zbývajícího času.

V programu jsem opět zachoval možnost doimplementovat v budoucnu řízení dalších generátorů a měřičů výkonu. K tomuto jsem využil dvě rozhraní, jedno pro generátory a druhé pro měřič výkonu. Rozložení tříd v programu je znázorněno na třídním diagramu

6.8. Jelikož by v tomto případě muselo se sériovým portem pracovat více tříd a obsluha by byla složitější, naimplementoval jsem proto sériový port do hlavního formuláře. Obsluhu portu poté řeším tak, že instance generátoru a měřiče výkonu vracejí metodami sjednocený příkaz pro sériový port. Příkaz je složen z části obsahující vysílaný příkaz na sériový port, očekávanou správnou odpověď a druh zařízení, které příkaz vysílá.



Obrázek 6.7: Uživatelské rozhraní zkušební aplikace.



Obrázek 6.8: Třídní diagram zkušební aplikace.

Pro obsloužení sériového portu volám metodu sendCommand a jako její parametr je výsledek metody generátoru nebo měřiče výkonu. V metodě pro obsluhu portu zkontroluji, zda je port připojený a zda již nějaká komunikace neběží. Pokud je podmínka vyhodnocena jako pravdivá, tak se aktivuje časovač, naplním proměnnou reprezentující

aktuální příkaz a nastavím příznak komunikace. Poté zkontroluji jaký typ generátoru je vybrán a odešlu data na sériový port. Celý proces je znázorněn ve výpisu 10.

```
sendCommand(generator.setFrequency(Convert.ToDouble(textBoxFstart.Text)));
sendCommand(powermeter.readPower());

private void sendCommand(Command cmd)
{
    if (serialPort1.IsOpen && comStatus != 1)
    {
        timerTimeout.Enabled = true;
        actualCommand = cmd;
        comStatus = 1;
        if (selectedGenerator != EDevice.SG2000) serialPort1.WriteLine(cmd.Commnd);
        else
        {
            string cmnd = cmd.Commnd + '\r';
            serialPort1.Write(cmnd);
        }
    }
}
```

Výpis 10: Obsluha sériového portu ve zkušební aplikace.

Detekce přijatých zpráv je založena na kódu ve výpisu 3 v kapitole 6.1. Pokud je rozeznána validní odpověď, tak je předána metodě ProcessingBuffer, kde je zpráva zkontrolována, zda obsahuje předpokládanou správnou odpověď. V případě, že je odpověď správná, tak se změní příznak stavu komunikace a iteruje proměnná počítající počet přijatých zpráv. Tento proces popisuje výpis kódu 11.

```
private void ProcessingBuffer(string command)
{
    if (actualCommand.Device == EDevice.PMM3000)
    {
        if (command.Contains(actualCommand.Reply))
        {
            receivedCount++;
            comStatus = 0;
        }
    }
    else if (actualCommand.Device == EDevice.SG2000)
    {
        if (command.Contains("OK"))
        {
            receivedCount++;
            comStatus = 0;
        }
        else
        {
            errorCount++;
            comStatus = -3;
        }
    }
}
```

Výpis 11: Zpracování přijatých zpráv.

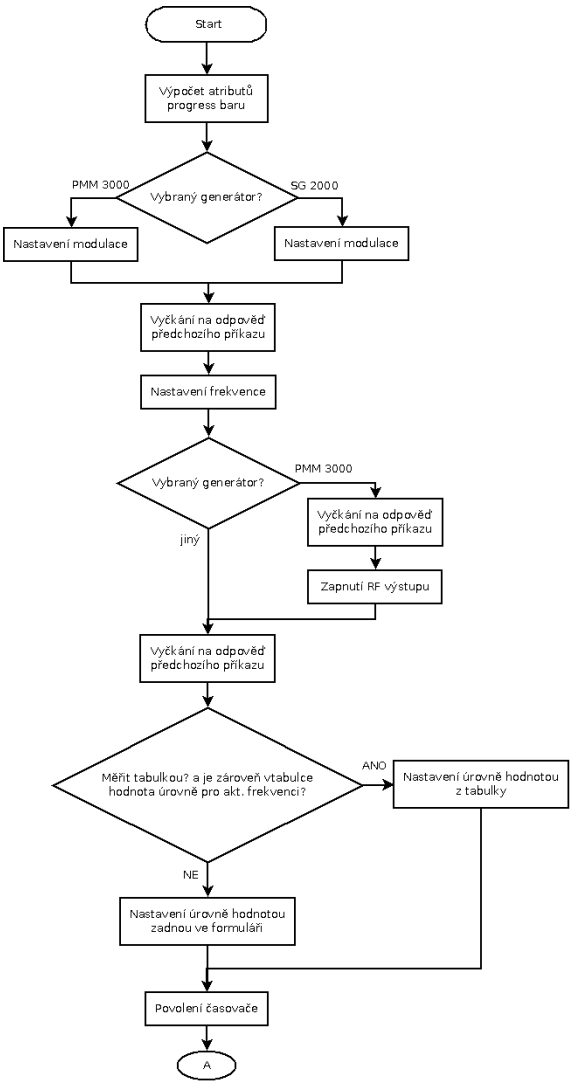
Tak jako v předchozích aplikacích byl problém s komunikací mezi vlákny, kdy uživatelské rozhraní běží v jiném vlákně než událost příjmu dat ze sériového portu. Tento problém jsem vyřešil stejně jako v aplikaci pro měřič výkonu, a to vytvořením delegáta. Tato část je popsána v kapitole 6.1 a výpisu kódu 3 v téže kapitole. Protože je nutné nastavovat více parametrů generátoru za sebou, tak jsem využil již vyvinutou metodu pro pozdržení vyslání dalšího příkazu v případě, že na předchozí příkaz ještě nedorazila odpověď. Metoda je zmíněna v kapitole 6.2 a výpisu kódu 9.

Prvotní návrh algoritmu pro automatizované nastavování parametrů generátoru se zdál bezproblémovým do doby, než jsem začal s ověřovacím měřením. Prvotní algoritmus vyčetl hodnotu naměřeného výkonu měřičem ihned po nastavení úrovně generátoru. Pokud byla změněna úroveň velkým skokem (například z +4 dBm na -20dBm), tak měřič naměřil špatnou hodnotu a až v dalším kroku měření při zachování stejné úrovně výkonu byla naměřena správná hodnota. Po experimentování s časem kroku jsem vyzkoušel vložit příkaz pro uspání vlákna mezi metodu nastavení frekvence a vyčtení hodnoty výkonu z měřiče. Tímto krokem se odstranil problém s naměřením špatné hodnoty výkonu. Problém je zapříčiněn nutností ustálení naměřené hodnoty v měřiči výkonu a je tedy dán konstrukcí měřiče. Po pár neúspěšných pokusech o řešení problému se mi naskytla myšlenka mírně poupravit původní algoritmus. Úprava spočívala v posunutí prvního a následujících vyčtení hodnot z měřiče do události přetečení časovače. Tímto jsem zajistil časový rozestup mezi nastavením frekvence a úrovně generátoru od vyčtení hodnoty výkonu z měřiče. Průběh algoritmu je popsán vývojovým diagramem 6.9 znázorňujícím začátek algoritmu až do spuštění časovače a obrázek 6.10, který znázorňuje vývoj algoritmu v události přetečení časovače.

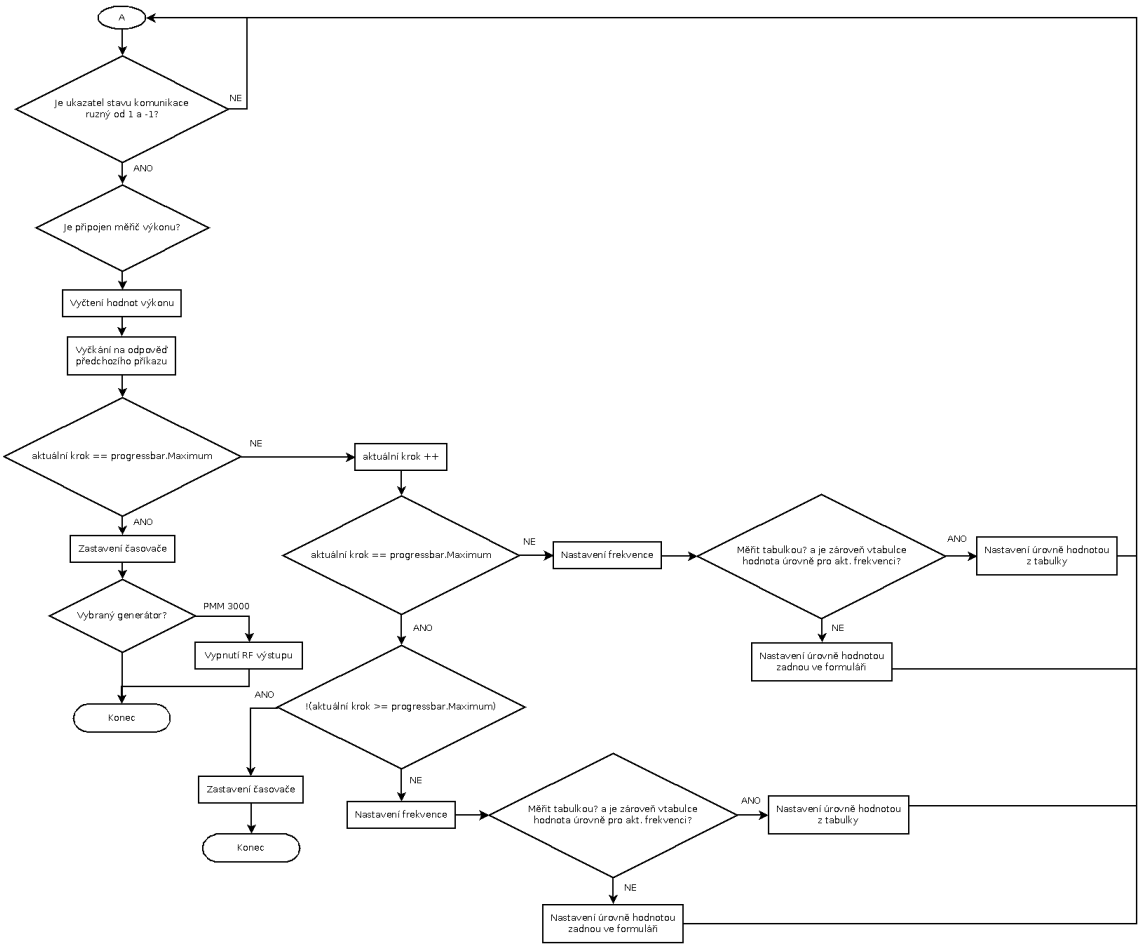
Frekvence, která má být nastavena v dalším kroku měření je vypočítána pomocí vzorce 6.6, kde x je velikost kroku v procentech a f_a je hodnota frekvence, která je aktuálně nastavena.

$$f = f_a * x \tag{6.6}$$

Parametry měření, jako počáteční frekvenci, koncovou frekvenci, velikost kroku v procentech, čas kroku a hodnotu úrovně, může uživatel editovat. Průběh měření lze zastavit a pokračovat pomocí tlačítka *Pauza*. Pokud neprobíhá měření, lze vyčíst hodnotu naměřeného výkonu pomocí tlačítka *Vyčti*.



Obrázek 6.9: Vývojový diagram ovládání komponent zkušební aplikace.



Obrázek 6.10: Vývojový diagram ovládání komponent zkušební aplikace.

7 Ověření funkcionality a výsledků práce

Výsledky a popis ověřování funkcionality galvanického oddělení a realizace dostupnosti přes počítačovou síť Ethernet jsou uvedeny v kapitole 3. Programové ovládání jsem ověřoval postupně během vývoje a také po jeho dokončení. Programové ovládání měřiče výkonu jsem nechal vyčítat 30 minut a nevyskytl se žádný problém během komunikace přes sériový port nebo při běhu programu. Programové ovládání generátorů jsem otestoval nastavením různých parametrů generátoru pro každý generátor. Nastavení parametrů probíhalo bez problémů a vyčítání parametrů taktéž. Programové ovládání pro zkoušku kompatibility EMC jsem otestoval propojením generátoru a měřiče výkonu koaxiálním kabelem. Provedl jsem několik zkušebních měřeních, které proběhly úspěšně.

Pro ověření výsledků měření novým programovým ovládáním a původního ovládaní, jsem propojil generátor a měřič výkonu. Provedl jsem nejprve měření původním programovým ovládáním, a poté novým ovládáním. Abych minimalizoval chyby, tak obě měření proběhly za sebou a na stejné sestavě se stejným rozložením komponent a kabelu. Měřil jsem na hodnotách frekvence 200 kHz, 2 MHz, 20 MHz a 200 MHz a hodnotách úrovně 10 dBm, 5 dBm, 0 dBm, -10 dBm, -20 dBm a -30 dBm. Naměřené hodnoty jsou zaznamenány v tabulkách 7.1 a 7.2.

Úroveň [dBm]	**Frekvence**			
	200 kHz	2 MHz	20 MHz	200 MHz
10	9,3	9,2	9,0	9,2
5	4,3	4,2	3,9	4,1
0	-1,2	-0,7	-0,6	-0,9
-10	-10,9	-11,5	-10,9	-11,8
-20	-20,9	-20,9	-20,9	-21,2
-30	-31,0	-31,2	-31,2	-31,5

Tabulka 7.1: Naměřené hodnoty původního ovládaní.

Úroveň [dBm]	**Frekvence**			
	200 kHz	2 MHz	20 MHz	200 MHz
10	9,5	9,3	9,0	9,1
5	4,6	4,4	4,0	4,1
0	-1,2	-1,3	-0,5	-1,5
-10	-11,1	-11,5	-10,8	-11,8
-20	-20,6	-21,0	-20,9	-21,3
-30	-30,7	-31,2	-31,2	-31,0

Tabulka 7.2: Naměřené hodnoty nového ovládaní.

Při porovnání tabulek je zřejmé, že odchylky mezi měřením původním nebo novým programem jsou minimální. Tyto odchylky jsou způsobeny nepřesností měření přístrojů, rozptyl hodnot se pohybuje v desetinách, což odpovídá tolerancím deklarovaných výrob-

cem. Program pracuje s již digitalizovanými daty, které jsou mu zasílány, a tedy nemůže ovlivnit přesnost. Mohu tedy tvrdit, že nové programové ovládaní pracuje stejně precizně, jako ovládaní původní.

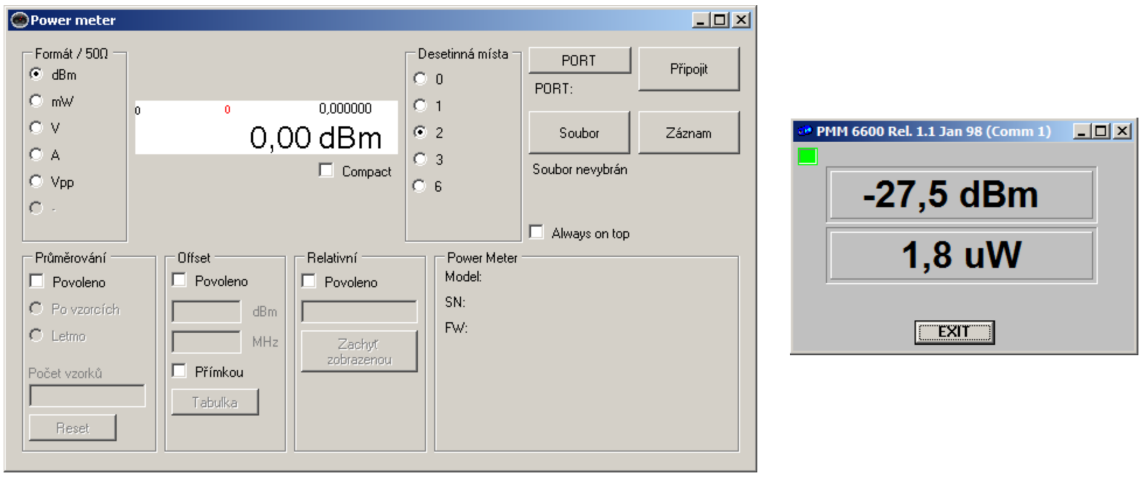
Spolehlivost a stabilita navrženého řešení je vyšší než u původního ovládaní, jelikož využívá současných programových prostředků určených k vývoji aplikací pro operační systémy Windows v dnešní době. Původní programové ovládaní bylo určeno pro 16bitové Windows 3.1.

Rychlost navrženého řešení je shodná s původním řešením, neboť pracovní rychlost postupu programu při zkoušce je pro automatizovaný režim dána normou, a to jeden krok za jednu sekundu. V manuálním režimu je programové ovládaní v podstatě programem typu HMI. Slouží tedy k ovládaní zařízení lidskou obsluhou a není tedy nutná extrémně velká odezva.

8 Závěr

Navrženým galvanickým oddělením se úspěšně zajistila ochrana částí řídícího a energetického obvodu před nechtěnými účinky vysokofrekvenční energie. Pomocí serveru sériové portu a virtuálního sériového portu se realizovala funkce dostupnosti přes počítačovou Ethernetovou síť. Následně vyvinuté programové ovládaní odstraňuje problémy původního již zastaralého programového ovládaní, které nové ovládaní rozšiřuje o některé funkce a možnosti. Navržené vlastní řešení oproti původnímu funguje bez problémů na moderních operačních systémech (Windows 7, Widnows 8), může využít jakýkoliv volný sériový port oproti původnímu řešení, které pracovalo pouze s porty COM 1 až COM 4, což může být v dnešní době problém, jelikož tyto porty bývají často obsazeny již jinými programy (BlueTooth a podobné).

Aplikace pro práci s měřičem výkonu je rozšířena oproti původní aplikaci o možnost zobrazení naměřeného výkonu ve více jednotkách, zobrazení informací o připojeném měřiči, funkci průměrování naměřených hodnot, funkci offsetu, funkci relativní hodnoty přičítané k naměřené hodnotě a možnosti záznamu naměřených hodnot do souboru ve formátu csv. Porovnání nového a starého uživatelského rozhraní je zobrazeno na obrázku 8.1.

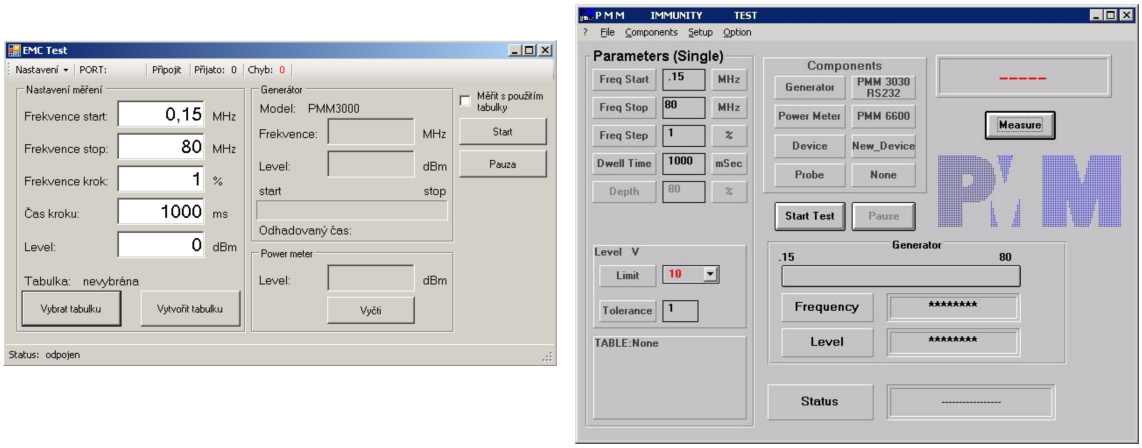


Obrázek 8.1: Porovnání uživatelských rozhraních měřiče výkonu.

Ovládací program generátoru se stal plnohodnotným nahrazením ovládacího panelu generátoru. Jelikož program nemá předlohu a nenahrazuje žádnou část původního programového ovládaní, tak je zcela vlastním řešením. Program umožňuje ovládaní dvou generátorů (PMM 3000 a SG 2000) a zdrojový kód je možné v budoucnu rozšířit o řízení dalších generátorů. Kromě nastavování parametrů generátoru disponuje program možností nastavení hodnoty úrovně výkonu v jednotkách decibel, volt a volt špička-špička. Uživatelské rozhraní je vyobrazeno na obrázku 6.5 v kapitole 6.2.

Programové ovládaní pro zkoušku kompatibility EMC byl hlavní důvod pro vývoj nového programu. Pomocí současných programových prostředků jsem odstranil problémy původního ovládaní zmíněné v prvním odstavci této kapitoly. Zároveň jsem nové ovlá-

daní rozšířil o možnost generování a exportu tabulky, která je zdrojem informací pro nastavování hodnoty úrovně k příslušné frekvenci. Rozhraní je také rozšířeno o výpočet odhadovaného času do konce měření. Zdrojový kód stejně jako u ovládacího programu pro generátor je připraven pro rozšíření o řízení dalších generátorů a měřičů výkonu. Porovnání nového a starého uživatelského rozhraní je zobrazeno na obrázku 8.2.



Obrázek 8.2: Porovnání uživatelských rozhraních ovládaní zkoušky kompatibility EMC.

Tato práce byla mým první větším projektem a první prací se sériovým portem. Předchozí zkušenosti s programováním v jazyce C# jsem měl pouze z vysoké školy, která mi poskytla pro tuto práci dobrý základ. Obsluhu sériového portu jsem považoval z počátku jako velmi obtížnou věc, ale postupem v práci a řešením problémů jsem se s obsluhou seznámil. V práci jsem si prohloubil znalosti jazyka C# a naučil se obsluhovat sériový port, jak práce v jazyce C#, tak práce se sériovým portem mě zaujala a v budoucnu bych rád opět řešil komunikaci přes sériovou linku. Kromě softwarové části práce jsem přišel do styku také s hardwarovou částí, a to realizací galvanického oddělení a funkce dostupnosti měřící sestavy přes počítačovou síť Ethernet, kde jsem si prakticky odzkoušel a využil poznatky z předmětů Optoelektronika a Počítačové sítě. Pro mne je tato práce velkým přínosem, jelikož jsem si prohloubil nabyté znalosti z vysoké školy a naučil jsem se technologie se kterými jsem doposud nepracoval.

Lukáš Stehlík

9 Literatura

[1] OLMR, Vít. HW server představuje - Sériová linka RS-232. *HW.cz — Vše o elektronice a programování* [online]. 2005 [cit. 2014-04-24]. Dostupné z: http://www.hw.cz/rozhrani/hw-server-predstavuje-seriova-linka-rs-232.html

[2] RS-485. *In: Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2014 [cit. 2014-04-24]. Dostupné z: http://cs.wikipedia.org/wiki/RS-485.

[3] STANĚK, Jan, ŘEHÁK, Jan RS 485 & 422. *HW.cz — Vše o elektronice a programování* [online]. 1998 [cit. 2014-04-24]. Dostupné z: http://www.hw.cz/teorie-a-praxe/dokumentace/rs-485-422.html

[4] POUCHA, Pavel. Přenos dat po linkách RS485 a RS422. *HW.cz — Vše o elektronice a programování* [online]. 1999 [cit. 2014-04-24]. Dostupné z: http://www.hw.cz/teorie-a-praxe/dokumentace/prenos-dat-po-linkach-rs485-a-rs422.html

[5] ŘEHÁK, Jan Virtuální sériový port zdarma. *HW.cz — Vše o elektronice a programování* [online]. 2003 [cit. 2014-04-24]. Dostupné z: http://www.hw.cz/navrh-obvodu/software/virtualni-seriovy-port-zdarma.html

[6] *DB9 Pinout* [online]. [cit. 2014-04-25]. Dostupné z: www.db9-pinout.com

[7] *Terminal* [online]. 2008, 2014 [cit. 2014-04-25]. Dostupné z: https://sites.google.com/site/terminalbpp/

[8] PMM COSTRUZIONI ELETTRONICHE CENTRO MISURE RADIOELETTRICHE S.R.L. *Operating Manual PMM 3000 wideband rf signal generator 10 kHz - 1000 MHz.* Italy, 2003.

[9] PMM COSTRUZIONI ELETTRONICHE CENTRO MISURE RADIOELETTRICHE S.R.L. *Operating Manual PMM 6600 rf power meter 10 kHz - 1 GHz.* Italy, 2003.

[10] ELSY SPOL S.R.O. *Signální generátor SG2000.*

[11] MOXA INC. *NPort IA5150/5250 Series Quick Installation Guide.* 4. vyd. 2008.

[12] TDK-LAMBDA *DSP10 Series Din Rail Power Technical Data Installation and Operation.*

[13] SALTEK S.R.O. *DA-275 DF, DA-275 DFI, DA-275 DF S Jednofázové přepěťové ochrany s VF filtrem třídy D – 3. stupeň.* 2010.

A Přílohy na DVD-ROM

Příloha 1

Text bakalářské práce v elektronické podobě.

Příloha 2

Zdrojové kódy programového ovládání.

Příloha 3

Zdrojové kódy aplikace simulující měřič výkonu.

Příloha 4

Diagramy a fotografie použité v bakalářské práci.

Příloha 5

Videonahrávky obsluhy zařízení vyvinutým programovým ovládáním.